

Faculté des Sciences Appliquées

Optimisation contrainte et non-contrainte par régions de confiance et avec approximations locales quadratiques

Jérôme Walmag

Thèse de doctorat en sciences appliquées
Promoteur : Éric J. M. Delhez

2010

L’homme raisonnable s’adapte au monde ; l’homme déraisonnable s’obstine à essayer d’adapter le monde à lui-même.

Tout progrès dépend donc de l’homme déraisonnable.

Georges Bernard Shaw

Remerciements

Il n’est un secret pour personne que la rédaction d’une thèse est un exercice qui met à contribution bien plus de personnes que celles dont le nom figure sur la couverture de l’ouvrage. De nombreuses personnes ont par leur participation — parfois inconsciente — contribué au bon déroulement de ce projet.

Je voudrais en tout premier lieu remercier Éric Delhez, à qui je dois cette aventure. Je voudrais d’abord le remercier pour sa confiance, son soutien, son attention, son intérêt pour l’évolution de mes recherches, ses bons conseils, ses qualités humaines et sa grande compréhension face à mes choix de vie qui ne furent pas exactement de ceux qui simplifient la vie d’un promoteur de thèse. Toutes ces qualités m’ont permis de mener ce travail à bon port. Les quelques moments passés à enseigner dans les séances de répétitions qu’il m’a confiées m’ont également permis de satisfaire ma passion pour l’éducation et la pédagogie. Pour tout cela, je tiens à lui exprimer toute ma gratitude.

Je remercie également l’Université de Liège, la faculté des Sciences Appliquées et le département A&M pour m’avoir permis de mener mes recherches dans un environnement stimulant. À ce titre, je tiens tout spécialement à remercier les collègues du groupe de Mathématiques générales pour toutes les discussions scientifiques approfondies mais aussi et surtout pour tous les bons moments passés à discuter de tout et de rien dans une ambiance détendue : merci à Patricia pour l’intérêt porté à mon travail, merci à Christophe pour les séances collectives de débogage. Merci également à Géraldine et Francine.

Je voudrais aussi remercier Caroline, Julien et Renaud, de vieux amis tous docteurs aujourd’hui, pour leur soutien moral : leurs encouragements m’ont aidé à garder le cap malgré les embûches.

Je réserve les dernières lignes à ceux qui ont vécu cette thèse en en subissant les conséquences les moins agréables. Merci à Delphine pour absolument tout mais encore pour tout le reste. Et merci aux deux petits bonshommes de deux et quatre ans qui m’ont d’ores et déjà appris bien plus que toutes les thèses du monde.

Table des matières

Table des figures

Liste des tableaux

Première partie

Introduction

Chapitre 1

Position du problème

Depuis la nuit des temps, l'homme optimise : que ce soit pour accélérer une technique de fabrication, pour minimiser le coût d'une construction ou pour ne pas payer plus d'impôt qu'il n'en faut. Il doit cependant tenir compte des contraintes externes pour parvenir à ses fins : accélérer une fabrication doit laisser inchangée la qualité du produit, minimiser le coût d'une construction ne doit pas pousser à l'inaction (la construction la moins chère étant celle qui n'existe pas) et le calcul de l'impôt doit répondre aux prescrits légaux et fiscaux.

Il convient d'abord de choisir sur base de quel(s) critère(s) un objet peut être considéré comme meilleur qu'un autre. La meilleure voiture est-elle la moins chère sur le marché ? Auquel cas l'épreuve de sélection est relativement simple : il suffit de comparer les prix des différents modèles. Mais la meilleure peut aussi être la plus sûre et, dans ce cas, une série d'expériences est nécessaire. Des dizaines d'autres critères sont possibles : espace disponible, vitesse, consommation de carburant, etc. Mais la définition de l'optimum peut aussi résulter d'une combinaison de ces différents critères : le rapport qualité/prix est la plus célèbre de ces combinaisons.

Les *techniques d'optimisation* sont des procédés systématiques permettant d'approcher, voire de trouver la technique de fabrication la plus rapide, le coût le plus bas ou l'impôt le plus juste. La technique d'optimisation la plus simple est sans conteste celle de l'*essai-erreur*. Il suffit de faire varier quelques paramètres et de conserver le candidat dès que celui-ci, tout en respectant les contraintes du problème, s'avère meilleur que le modèle le plus performant actuellement disponible. Avec un peu d'habitude, d'intuition ou de chance, une amélioration peut être obtenue. Mais s'agit-il vraiment d'un *optimum* ?

Les techniques plus sophistiquées, dont celles que présentent ce travail, ont recours à une *formulation mathématique*.

1.1 Formulation mathématique.

La formulation mathématique d'un problème d'optimisation doit en identifier l'objectif, les variables et les contraintes. Cette formulation est la plus importante — et souvent la plus difficile — des étapes. De cette formulation, le présent travail ne parle pas : nous partirons d'une formulation mathématique générale.

La première étape consiste à identifier les *variables* d'un problème. Les variables sont les éléments sur lesquels nous avons la possibilité d'agir directement pour en modifier les valeurs. Les variables sont généralement amalgamées dans un vecteur x.

L'étape suivante consiste à définir la *fonction objectif* $f(x)$. La fonction objectif mesure la quantité à minimiser ou à maximiser. Sauf mention contraire, dans le reste de ce travail, nous nous concentrerons sur les problèmes de *minimisation* de $f(x)$, un problème de maximisation pouvant être facilement converti en cherchant à minimiser l'opposé de la fonction $f(x)$.

Il convient enfin d'identifier les *fonctions de contraintes* $c_j(x)$. Les contraintes peuvent être de deux natures : nous avons, d'une part, les contraintes d'égalité et, d'autre part, les contraintes d'inégalité.

Plus formellement, le problème général que nous traitons s'écrit

$$\begin{aligned} \text{minimiser} \quad & f(x), \\ \text{s.c.} \quad & c_j(x) = 0 \quad \text{pour } j \in \mathcal{E}, \\ & c_j(x) \leq 0 \quad \text{pour } j \in \mathcal{I} \end{aligned} \tag{1.1}$$

où $\mathcal{E}$ et $\mathcal{I}$ sont, respectivement, les ensembles disjoints des indices des contraintes d'égalité et d'inégalité. Les fonctions f et c_j sont supposées continûment dérivables. On désigne par Ω, l'*ensemble admissible*, le sous-ensemble de $\mathbb{R}^n$ où les contraintes sont satisfaites, *i.e.*

$$\Omega = \left\{ x \in \mathbb{R}^n \mid c_j(x) = 0 \text{ pour } j \in \mathcal{E} \text{ et } c_j(x) \leq 0 \text{ pour } j \in \mathcal{I} \right\}. \tag{1.2}$$

Pour certains problèmes, les variables n'ont un sens qu'à la condition d'être choisies dans un ensemble de valeurs discrètes. Il s'agit là d'*optimisation discrète*, à opposer à l'*optimisation continue*. Le présent travail ne traitera pas de l'optimisation discrète.

1.2 Conditions d'optimalité.

Les solutions les plus intéressantes d'un problème d'optimisation sont les *minima globaux*, c'est à dire l'ensemble des arguments x^* pour lequel la fonction f atteint sa plus petite valeur dans l'ensemble admissible, *i.e.*

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega. \tag{1.3}$$

Le minimum global est généralement difficile à trouver car les valeurs des fonctions $f(x)$ et $c_j(x)$ ne sont généralement pas connues en tous les points de l'ensemble admissible. La plupart des algorithmes se contentent donc de trouver un *minimum local*.

On dit que la fonction objectif présente un *minimum local* en un point x^* si la fonction $f(x^*)$ est la plus petite valeur dans un voisinage de ce point[1], *i.e.* x^* est un minimum local s'il existe un voisinage $V(x^*)$ tel que

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega \cap V(x^*). \tag{1.4}$$

Le minimum local est dit *strict* si l'inégalité peut être remplacée par une inégalité stricte.

Heureusement, cette définition ne constitue pas la seule manière de déterminer si un point x^* est oui ou non un minimum local. Il serait en effet impossible d'explorer tout les points du voisinage de x^* pour être sûr que la fonction objectif n'est pas supérieure à $f(x^*)$. Lorsque le problème est suffisamment régulier, des moyens beaucoup plus efficaces existent. Par exemple, lorsque la fonction f est deux fois continûment dérivable, les critères d'optimalité sont bien connus (voir, par exemple, Fletcher [31]). Dans un problème non-contraint, il suffit que le *gradient* de f au point x^* soit nul

$$\nabla_x f(x^*) = 0 \tag{1.5}$$

et que son *Hessien* (aussi appelé *matrice hessienne*) $\nabla_{xx} f(x^*)$ soit définie positive pour que x^* soit un minimum local de f.

La littérature fait aussi grand usage des *conditions nécessaires* d'optimalité. La condition du premier ordre est tout simplement

$$\nabla_x f(x^*) = 0. \tag{1.6}$$

Les points répondant à cette propriété sont appelés *points stationnaires* ou *points critiques du premier ordre*. La condition nécessaire du second ordre se formule

$$\nabla_{xx} f(x^*) \text{ semi-définie positive.} \tag{1.7}$$

Pour les problèmes contraints, l'établissement d'une condition nécessaire demande de faire quelques hypothèses sur la régularité des contraintes. Ces conditions de régularité sont généralement connues sous le nom de *qualification des contraintes*. Il existe plusieurs hypothèses de qualification des contraintes ; nous n'aborderons que la plus simple et la plus connue (voir, par exemple, Nocedal et Wright [80]) : la *condition d'indépendance linéaire de qualification des*

[1] Un voisinage $V(x)$ de x est un ensemble contenant un ouvert contenant x.

contraintes. Pour satisfaire cette condition, il suffit que les gradients $\nabla_x c_j(x^*)$ des contraintes actives[2] au point x^* soient linéairement indépendants. Dans ce cas, nous avons les célèbres conditions nécessaires du premier ordre de Karush–Kuhn–Tucker [59]

$$\begin{aligned} \nabla_x f(x^*) + \sum_{j \in \mathcal{E} \cup \mathcal{I}} \lambda_j^* \nabla_x c_j(x^*) &= 0, \\ c_j(x^*) &= 0 \quad \text{pour } j \in \mathcal{E}, \\ c_j(x^*) &\leq 0 \quad \text{pour } j \in \mathcal{I}, \\ \lambda_j^* &\geq 0 \quad \text{pour } j \in \mathcal{I}, \\ \lambda_j^* c_j(x^*) &= 0 \quad \text{pour } j \in \mathcal{E} \cup \mathcal{I}. \end{aligned} \tag{1.8}$$

Les paramètres λ_j^* sont les *multiplicateurs de Lagrange* du problème.

En règle générale, il n'est pas possible de déterminer si un minimum local est le minimum global du problème.

1.3 Taux de convergence.

Les différentes techniques d'optimisation se distinguent bien entendu par leurs performances. Une des mesures de la performance d'un algorithme est le *taux de convergence* ; il s'agit d'une mesure de la vitesse de convergence. Les deux taux de convergence les plus utilisés sont définis comme suit (voir, par exemple, Nocedal et Wright [80]).

Soit une suite d'itérés $\{x^{(k)}\}$ de $\mathbb{R}^n$ qui converge vers x^*. On dit que le taux de convergence est *linéaire* s'il existe une constante $r \in]0,1[$ telle que

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} \leq r \quad \text{pour } k \text{ suffisamment grand.} \tag{1.9}$$

Cela signifie que l'écart entre l'itéré et la solution décroît au moins, à chaque itération, d'un facteur constant r. La convergence est *superlinéaire* si

$$\lim_{k \to \infty} \frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|} = 0. \tag{1.10}$$

Une convergence *quadratique* se définit quant à elle par

$$\frac{\|x^{(k+1)} - x^*\|}{\|x^{(k)} - x^*\|^2} \leq M \quad \text{pour } k \text{ suffisamment grand,} \tag{1.11}$$

où M est une constante positive. Cela signifie que l'écart avec la solution décroît quadratiquement au fil des itérations.

[2] Une contrainte $c_j(x)$ est dite *active* au point x^* si $c_j(x^*) = 0$.

1.4 Objet et apport de ce travail.

Ce travail porte sur une classe particulière de techniques d'optimisation : il s'intéresse aux problèmes non-linéaires, contraints et non-contraints. Les méthodes envisagées sont *itératives* : elles construisent, au fur et à mesure des itérations, une suite $\{x^{(k)}\}$ qui — du moins nous l'espérons — convergera vers un minimum local x^*. Un point de départ $x^{(0)}$ pour les paramètres à optimiser est arbitrairement choisi et la fonction objectif $f(x^{(0)})$ ainsi que les fonctions de contraintes $c_j(x^{(0)})$ sont évaluées. Pour chaque itéré $x^{(k)}$ ces mêmes fonctions seront également évaluées. Notre travail s'inscrit dans le paradigme où le temps nécessaire pour évaluer celles-ci s'avèrent extrêmement long par rapport aux calculs nécessaires à l'algorithme lui-même. Plus formellement, cette hypothèse peut s'exprimer comme suit.

Hypothèse 1.1 *Le temps nécessaire au calcul de l'itéré $x^{(k+1)}$ à partir de la valeur de la fonction objectif $f(x^{(k)})$, des fonctions de contraintes $c_j(x^{(k)})$ et/ou de leurs dérivées au point $x^{(k)}$ est négligeable par rapport au temps nécessaire à l'évaluation de ces dernières.*

Ceci signifie que notre priorité, en terme de performance, est à la diminution du nombre total d'itérations engendrées par l'algorithme. Ce paradigme présente l'avantage que la mesure de performance ne dépend ni du problème envisagé ni de la puissance de calcul utilisée. Cette hypothèse est réaliste et utile : pensons par exemple aux modèles numériques complexes utilisés en météorologie ou en aéronautique. Comme toutes les hypothèses, celle-ci à ses limites : en particulier lorsque nous avons affaire à des problèmes de grande taille puisque le nombre d'opérations de calcul causé par l'algorithme lui-même croît généralement exponentiellement avec le nombre de paramètres de la fonction à optimiser.

La première partie du travail, incluant le présent chapitre, fait office d'introduction générale aux méthodes par régions de confiance et à l'utilisation d'approximations locales quadratiques. Le chapitre 2 porte sur différentes techniques de globalisation et sur celle qui nous intéressera tout au long de ce travail : la méthode dite *des régions de confiance*. En quelques mots, elle consiste à considérer que les fonctions $f(x)$ et $c_j(x)$ peuvent être remplacées par une *approximation locale*, plus facile à utiliser, dans une certaine zone de confiance autour de l'itéré courant $x^{(k)}$. La question du choix de l'approximation locale la plus appropriée est abordée au chapitre 3. Plusieurs types d'approximations locales sont développées en détail mais l'essentiel de notre travail se concentre sur les approximations quadratiques. Les approximations de Newton, Newton modifiées, quasi-Newton sont abondamment utilisées dans la suite du travail. Le chapitre 4 aborde, avec de nombreux détails, les aspects théoriques de convergence globale des méthodes

par régions de confiance. Les hypothèses nécessaires à l'établissement des théorèmes de convergence vers des points critiques du premier et du second ordre y sont exposées. Nous y traitons également de la forme des régions de confiance et des critères de convergence pour des problèmes non-différentiables.

Le premier de ces problèmes fait l'objet du chapitre 5 : il aborde la question de l'*identification paramétrique* (ou la *calibration*) d'un *modèle dynamique* par rapport à des mesures expérimentales. Cette identification peut être exprimée sous la forme d'un problème d'optimisation non-contraint et l'observabilité des paramètres à identifier peut être discutée au moyen d'une *expérience jumelle*. Mais c'est surtout la question de la différentiation de la fonction objectif (et donc du modèle sous-jacent) qui est discutée en détail : son coût en ressources informatiques est en effet un facteur-clé pour mener à bien la calibration.

Le chapitre 6 développe en détail l'algorithme Trust et ses différentes variantes pour l'optimisation non-contrainte. Il s'agit là d'une implémentation particulière de la méthode des régions de confiance. Elle utilise des approximations locales quadratiques de type quasi-Newton. Nous abordons, dans ce chapitre, la question du choix de la méthode de différentiation de la fonction objectif et son influence sur la vitesse de convergence de l'algorithme d'optimisation servant à la calibration. Un exemple éloquent sur un simple modèle proie-prédateur de Lotka–Volterra est ensuite présenté et discuté. C'est un des apports majeurs de ce travail qui fait par ailleurs l'objet d'une publication (Walmag et Delhez [101]).

Le chapitre 7 est le deuxième apport majeur de ce travail. Il présente une stratégie nouvelle de mise à jour du *rayon de confiance*. Le rayon de confiance est un paramètre des méthodes par régions de confiance ; il indique l'étendue de la dite région autour de l'itéré $x^{(k)}$. Traditionnellement, l'étendue de cette zone de confiance est maintenue constante si elle mène à des résultats satisfaisant, diminuée si ceux-ci sont médiocres et augmentée s'ils sont bons. Nous entendons par « résultat », la diminution effective de la fonction objectif. La nouveauté introduite ici invite à se méfier des « trop bons » résultats, c'est-à-dire des itérations menant à une réduction de la fonction objectif bien plus importante que prévue par l'approximation locale. Dans ce cas, la logique proposée est de maintenir quasi-constant le rayon de confiance. Cette simple précaution améliore les performances de l'algorithme et cette amélioration est considérable lorsqu'elle est cumulée avec une stratégie intelligente de mise à jour de l'approximation quadratique. Les profils de performances des différentes stratégies ont été comparés et validés sur nombre de problèmes-tests et cette partie du travail fait également l'objet d'une publication (Walmag et Delhez [100]).

La troisième partie du travail aborde la question de l'optimisation contrainte. Le chapitre 8 trace les grandes lignes des *méthodes quadratiques séquentielles* (SQP) conjuguées avec une globalisation par régions de confiance : cette technique bien connue, et plus particulièrement les *corrections du second ordre*, ser-

viront de base au chapitre suivant.

Le chapitre 9 présente un algorithme complet et efficace de résolution d'un sous-problème convexe *quadratique à contraintes quadratiques*. Cette méthode originale se base sur une fonction de pénalité de type ℓ_1 : celle-ci présente la particularité de ne pas être différentiable. L'algorithme tire profit d'une décomposition de l'espace des variables x en trois sous-espaces orthogonaux : un premier permettant de gérer des contraintes de bornes, un deuxième dans lequel la fonction objectif est continûment dérivable et un troisième où elle présente des « cassures de pente ». Le principe du nouvel algorithme est de construire une stratégie de recherche de direction de descente successivement sur ces trois sous-espaces. Les performances de cet algorithme sont encore améliorées par l'implémentation d'un *mode rapide* qui tire profit des corrections du second ordre utilisées dans les méthodes SQP et par une méthode de recherche unidimensionnelle particulièrement bien adaptée.

Le chapitre 10 aborde l'utilisation de l'algorithme UVQCQP élaboré au chapitre précédent dans le cadre de la résolution de problèmes non-linéaires et non-convexes. L'approche envisagée est celle des *algorithmes séquentiels de programmation quadratique à contraintes quadratiques* : ses avantages sont abordés rapidement par l'entremise de quelques exemples. Quelques tests numériques ont été effectués et sont encourageants.

Enfin, la quatrième et dernière partie (chapitre 11), tire les conclusions et dégage les perspectives futures.

Chapitre 2

Méthodes de globalisation en optimisation

Le domaine de l'optimisation non-contrainte a vu l'émergence de nombreuses méthodes et algorithmes. Il est vain de vouloir tous les présenter aussi nous contenterons-nous, dans les deux chapitres qui suivent, d'expliciter quelques caractéristiques principales des différentes familles d'algorithmes.

Ce chapitre traite d'abord des méthodes dites *basées sur le gradient* ; il aborde ensuite succinctement la famille des méthodes dites *méta-heuristiques*.

Les méthodes basées sur le gradient sont itératives. Elles démarrent en un point de départ $x^{(0)}$ et génèrent une suite $\{x^{(k)}\}$ qui est interrompue lorsque plus aucun progrès ne peut être fait ou lorsqu'une solution semble avoir été approchée avec une précision suffisante. Pour décider comment passer d'un itéré $x^{(k)}$ au suivant, ces méthodes se basent sur des informations locales sur f obtenues au point $x^{(k)}$ et éventuellement aux points précédents : elles construisent une *approximation locale* $m^{(k)}(x)$ plus ou moins sophistiquée de la fonction objectif (plus facile à traiter que la fonction objectif elle-même[1]) qui est utilisée pour résoudre un problème local et trouver un nouvel itéré $x^{(k+1)}$ pour lequel la valeur de la fonction objectif sera moindre. D'une façon ou d'une autre, ces algorithmes doivent revenir régulièrement à la *véritable* fonction objectif $f(x)$ et ne peuvent indéfiniment traiter avec l'ersatz qu'est $m^{(k)}(x)$ au risque de se fourvoyer dangereusement. Cette étape qui consiste à repasser du problème local vers le problème global est appelée *globalisation*. Deux méthodes de globalisation sont abordées en détail dans les section suivantes : l'approche par *recherche linéaire* et celle par *régions de confiance*. Une troisième méthode de globalisation dite du *point proximal* sera rapidement évoquée dans un souci de complétude de l'exposé.

[1] Le choix de ces approximations locales est traité au chapitre 3.

2.1 Globalisation par recherche linéaire.

Dans l'approche par recherche linéaire, une *direction de descente* $d^{(k)}$ est construite et une recherche est effectuée le long de cette direction en partant de $x^{(k)}$ pour trouver un nouvel itéré dont la valeur de la fonction objectif est plus petite. La direction $d^{(k)}$ est une direction de descente s'il existe $C > 0$ tel que

$$f(x^{(k)} + \varepsilon d^{(k)}) < f(x^{(k)}), \forall \varepsilon < C, \tag{2.1}$$

ce qui devient simplement, dans le cas où f est dérivable en $x^{(k)}$,

$$d^{(k)T} \nabla_x f(x^{(k)}) \leq 0. \tag{2.2}$$

La distance dont l'algorithme avance dans la direction $d^{(k)}$ est appelée le *pas de progression* et peut être trouvée en résolvant un problème unidimensionnel de *recherche linéaire* à l'itération k. Il s'agit de calculer — éventuellement de façon approchée — le pas $\xi^{(k)}$ à parcourir le long de la direction de descente $d^{(k)}$ afin de minimiser la fonction objectif, *i.e.* trouver

$$\xi^{(k)} = \arg\min_{\xi} \psi(\xi) \tag{2.3}$$

où $\psi(\xi) = f(x^{(k)} + \xi d^{(k)})$.

La nouvelle approximation de la solution sera alors obtenue par la mise à jour

$$x^{(k+1)} = x^{(k)} + \xi^{(k)} d^{(k)}. \tag{2.4}$$

En résolvant exactement le problème (2.3), nous tirerions un bénéfice maximal de la direction $d^{(k)}$. Mais une minimisation exacte est cependant très coûteuse et n'est généralement pas nécessaire. Il est donc très fréquent de voir les méthodes de recherche linéaire dispensées de trouver le minimum exact dans la direction de descente. Le plus fréquemment, la recherche est arrêtée dès qu'une diminution suffisante de la fonction objectif est constatée (voir par exemple [80]). On demande généralement le respect de la *condition d'Armijo*

$$f(x^{(k)} + \xi^{(k)} d^{(k)}) \leq f(x^{(k)}) + c_1 \xi^{(k)} d^{(k)T} \nabla_x f(x^{(k)}) \tag{2.5}$$

pour une constante donnée $0 < c_1 < 1$. En pratique, cette constante c_1 est généralement choisie assez petite (de l'ordre de 10^{-4}). Cette condition de décroissance n'est cependant pas suffisante pour garantir une progression satisfaisante de l'algorithme ; la condition (2.5) autorise en effet des pas de progression extrêmement petits. Pour s'en prémunir, on introduit la *condition de courbure*

$$d^{(k)T} \nabla_x f(x^{(k)} + \xi^{(k)} d^{(k)}) \geq c_2 d^{(k)T} \nabla_x f(x^{(k)}) \tag{2.6}$$

avec $c_1 < c_2 < 1$. Pour éviter d'obtenir des pas de progression trop différents de (2.3), la condition (2.6) est parfois modifiée selon

$$|d^{(k)T}\nabla_x f(x^{(k)} + \xi^{(k)} d^{(k)})| \leq c_2 \, |d^{(k)T}\nabla_x f(x^{(k)})|. \tag{2.7}$$

Les conditions (2.5) et (2.6) sont appelées les *conditions de Wolfe* tandis que (2.5) et (2.7) sont les *conditions fortes de Wolfe*. Notons qu'il est toujours possible de trouver un minimum $\xi^{(k)}$ répondant aux conditions de Wolfe si $d^{(k)}$ est une direction de descente. Comme nous le verrons plus tard, ces conditions peuvent s'avérer particulièrement utiles dans le cadre des méthodes de type quasi-Newton (voir section 3.2.3).

Parallèlement aux conditions de Wolfe, les *conditions de Goldstein* remplissent une fonction similaire

$$f(x^{(k)} + \xi^{(k)} d^{(k)}) \geq f(x^{(k)}) + (1-c)\,\xi^{(k)} d^{(k)T}\nabla_x f(x^{(k)}) \tag{2.8}$$
$$f(x^{(k)} + \xi^{(k)} d^{(k)}) \leq f(x^{(k)}) + c\,\xi^{(k)} d^{(k)T}\nabla_x f(x^{(k)}) \tag{2.9}$$

avec $0 < c < \frac{1}{2}$. Il n'est cependant pas possible de garantir l'existence d'un minimum $\xi^{(k)}$ répondant à ces conditions.

Il existe plusieurs méthodes de minimisation pour le problème unidimensionnel (2.3) que nous classerons ici suivant le nombre de points où une évaluation de la fonction et, parfois, de ses dérivées est nécessaire. Dans la suite de l'exposé, nous supposerons que la fonction $\psi(\xi)$ est deux fois continûment dérivable.

Le lecteur cherchant des précisions à propos des méthodes développées dans cette section est invité à consulter l'ouvrage de Stoer et Bulirsch [95] par exemple.

2.1.1 Méthodes à un point.

Les méthodes itératives à un point cherchent à annuler la dérivée première en utilisant les informations disponibles en un point. Un processus itératif de type Newton–Raphson peut être utilisé afin d'identifier un zéro de la dérivée de $\psi(\xi)$. Considérons l'approximation de Taylor limitée au premier ordre de cette dérivée autour d'un point ξ

$$\psi'(\xi^+) \simeq \psi'(\xi) + \psi''(\xi)\,(\xi^+ - \xi). \tag{2.10}$$

Annulant la valeur de la dérivée au point ξ^+ qui deviendra l'itéré suivant, on obtient la formule d'actualisation du problème de recherche linéaire

$$\xi^+ = \xi - \frac{\psi'(\xi)}{\psi''(\xi)}. \tag{2.11}$$

Si $\psi(\xi)$ est deux fois différentiable et si sa dérivée seconde $\psi''(\xi)$ est continue au sens de Lipschitz[2] dans un voisinage d'une solution ξ^*, on peut montrer que cette méthode présente un taux de convergence quadratique et que sa convergence est assurée vers ξ^* si le point de départ est suffisamment proche de ξ^* (voir [80] par exemple). Outre le fait qu'il est difficile de s'assurer que le point de départ est « suffisamment proche » de la solution ξ^*, cette méthode nécessite le calcul de la dérivée seconde de la fonction objectif le long de la direction de descente. La dérivée seconde n'est malheureusement pas toujours disponible.

La *méthode de la corde* s'inspire de la méthode de Newton–Raphson et approche l'inverse de la dérivée seconde de $\psi(\xi)$ par une constante m

$$\psi''(\xi) \simeq \frac{1}{m} \tag{2.13}$$

et la formule d'actualisation est dès lors

$$\xi^+ = \xi - m\psi'(\xi) \tag{2.14}$$

L'ordre de convergence est affecté par cette approximation, il passe de quadratique à linéaire.

Ces deux méthodes à un point ont cependant un défaut majeur : leur convergence globale vers une solution ξ^* n'est assurée que dans des conditions plutôt restrictives, ce qui en fait de piètres instruments de *globalisation* pour des fonctions objectifs tout à fait générales.

2.1.2 Méthodes à deux points.

Les méthodes à deux points utilisent les informations en deux points et nécessitent dès lors une procédure spécifique pour obtenir les deux premiers points.

Tout d'abord, la *méthode de la corde classique* s'inspire de la méthode de la corde mais approche la dérivée seconde de $\psi(\xi)$ par une différence finie

$$\psi''(\xi) \simeq \frac{\psi'(\xi) - \psi'(\xi^-)}{\xi - \xi^-} \tag{2.15}$$

où ξ désigne l'itéré courant et ξ^- celui qui le précède. Le processus itératif complet s'écrit donc

$$\xi^+ = \xi - \frac{\xi - \xi^-}{\psi'(\xi) - \psi'(\xi^-)}\,\psi'(\xi) \tag{2.16}$$

[2] Pour rappel, une fonctions $f(x) : \mathcal{D} \subseteq \mathbb{R} \to \mathbb{R}$ est continue au sens de Lipschitz s'il existe une constante κ telle que, pour tout $x, y \in \mathcal{D}$,

$$|f(x) - f(y)| \leq \kappa |x - y|. \tag{2.12}$$

De façon assez surprenante, on peut montrer que le taux de convergence de ce processus est le *nombre d'or* $\frac{1+\sqrt{5}}{2}$. Malheureusement, la convergence globale ne peut être établie, cette méthode est donc rarement utilisée telle quelle.

Cette difficulté à propos de la convergence globale peut être résolue en utilisant une méthode avec *intervalle*. L'idée de base est de trouver un intervalle $[\xi_1, \xi_2]$ tel que la dérivée soit négative au point ξ_1 et positive au point ξ_2. Vu le caractère continu de la fonction ψ, cette condition assure la présence d'un minimum à l'intérieur de l'intervalle. L'algorithme tente alors de réduire celui-ci, tout en conservant la condition sur les dérivées aux bornes de chaque nouvel intervalle. Pour ce faire, un nouveau point appartenant à l'intervalle considéré, noté ξ_3, est calculé selon une formule du type

$$\xi_3 = \xi_2 - \rho(\xi_2 - \xi_1). \tag{2.17}$$

Le calcul de la valeur du paramètre $\rho \in [0,1]$ à partir des résultats obtenus aux extrémités de l'intervalle $[\xi_1, \xi_2]$ dépend de la méthode de résolution choisie. Ensuite, après calcul de la dérivée en ce point, l'algorithme réduit l'intervalle. Si $\psi'(\xi_3)$ est négatif, l'intervalle $[\xi_3, \xi_2]$ est utilisé pour l'itération suivante et, dans le cas contraire, l'algorithme utilise $[\xi_1, \xi_3]$.

Parmi les méthodes utilisant un intervalle, la *méthode de la bisection* est la plus simple : à chaque itération l'intervalle obtenu est divisé en deux parties égales ($\rho = 1/2$). L'équation (2.17) devient alors

$$\xi_3 = \frac{\xi_1 + \xi_2}{2}. \tag{2.18}$$

Cette méthode n'utilise cependant aucune des informations sur les valeurs des dérivées $\psi(\xi_1)$ et $\psi(\xi_2)$ alors que celles-ci sont disponibles et ont été évaluées au cours des itérations précédentes. La convergence peut donc être améliorée sans coût supplémentaire, c'est la raison pour laquelle la méthode de la bisection n'est que rarement utilisée en pratique.

La méthode *regula falsi* se base sur la formule d'actualisation de la méthode à un point de Newton–Raphson, considérée au point ξ_2. Cependant, afin d'éviter le calcul de la dérivée seconde $\psi''(\xi_2)$, celle-ci est approchée par différences finies sous la forme

$$\psi''(\xi_2) \simeq \frac{\psi'(\xi_2) - \psi'(\xi_1)}{\xi_2 - \xi_1} \tag{2.19}$$

pour obtenir la formule de mise à jour

$$\xi_3 = \xi_2 - \frac{\psi'(\xi_2)}{\psi'(\xi_2) - \psi'(\xi_1)} (\xi_2 - \xi_1). \tag{2.20}$$

Il est possible de montrer que, si pour une itération quelconque, la dérivée troisième de la fonction objectif existe et est positive dans l'intervalle $[\xi_1, \xi_2]$, le taux de convergence de cette méthode est linéaire (voir [95]).

La méthode à deux point la plus populaire est sans conteste celle de l'*interpolation cubique* qui approche la fonction $\psi(\xi)$ par un polynôme du troisième degré en ξ à partir des valeurs de $\psi(\xi_1)$, $\psi(\xi_2)$, $\psi'(\xi_1)$ et $\psi'(\xi_2)$ calculées lors des itérations précédentes. La nouvelle approximation du pas optimal ξ_3 est alors choisie comme le minimum de cette interpolation cubique. Celle-ci est calculée grâce à l'équation (2.17) où la valeur de ρ est donnée par

$$\rho = \frac{\psi'(\xi_2) + S + R}{\psi'(\xi_2) - \psi'(\xi_1) + 2S} \tag{2.21}$$

où

$$S = 3\,\frac{\psi(\xi_1) - \psi(\xi_2)}{\xi_2 - \xi_1} + \psi'(\xi_1) + \psi'(\xi_2) \tag{2.22}$$

$$R = \sqrt{S^2 - \psi'(\xi_1)\psi'(\xi_2)}. \tag{2.23}$$

Cette méthode est très utilisée en raison de son taux de convergence quadratique et de sa propriété de convergence globale indépendante du point de départ (voir [82, 95]).

2.1.3 Méthode à trois points.

Considérons maintenant un ensemble de trois valeurs croissantes du pas, soit $\xi_1 < \xi_2 < \xi_3$, où les valeurs de la fonction objectif $\psi(\xi)$ sont évaluées. Le minimum ξ_4 de l'*interpolation quadratique* passant par ces trois points est ajouté à l'ensemble ordonné

$$\xi_4 = \frac{1}{2}\,\frac{b_{23}\,\psi(\xi_1) + b_{31}\,\psi(\xi_2) + b_{12}\,\psi(\xi_3)}{a_{23}\,\psi(\xi_1) + a_{31}\,\psi(\xi_2) + a_{12}\,\psi(\xi_3)} \tag{2.24}$$

où $a_{ij} = \xi_i - \xi_j$ et $b_{ij} = \xi_i^2 - \xi_j^2$. Il convient alors d'exclure de l'ensemble ordonné ξ_1 ou ξ_3 (celui dont la valeur de la fonction objectif est la plus grande) et de recommencer l'opération avec les trois valeurs restantes. Cette méthode à trois points présente l'avantage d'avoir un taux de convergence quadratique et de ne pas nécessiter le calcul des dérivées.

Dans le cas de fonctions non-convexes, il peut arriver que ξ_4 n'appartienne pas à l'intervalle $[\xi_1, \xi_3]$. Dans ce cas l'algorithme est redémarré avec, par exemple,

les trois points

$$\begin{aligned}
&(\xi_1, \frac{\xi_1+\xi_2}{2}, \xi_2) \qquad \text{si } \psi(\xi_1) < \psi(\xi_3) \\
&(\xi_2, \frac{\xi_2+\xi_3}{2}, \xi_3) \qquad \text{sinon.}
\end{aligned} \tag{2.25}$$

2.2 Globalisation par régions de confiance.

Les méthodes d'optimisation par régions de confiance se basent sur une idée simple : à chaque itération, l'*approximation locale*[3] $m^{(k)}(x)$ est considérée comme fiable dans un domaine de validité déterminé, une région de confiance, dont la taille est adaptée au fur et à mesure des itérations. Moyennant quelques hypothèses, la convergence globale de cette approche vers un minimum local peut être rigoureusement établie (voir Conn, Gould et Toint [20]).

À chaque itération, l'algorithme définit une approximation locale $m^{(k)}(x)$ dont le but est d'approcher la fonction objectif dans une région de confiance

$$\mathcal{B}^{(k)} = \left\{ x \in \mathbb{R}^n : \; \|x - x^{(k)}\|_k \leq \Delta^{(k)} \right\} \tag{2.26}$$

où $\Delta^{(k)}$ est le *rayon de confiance* et où $\|\cdot\|_k$ est une norme dépendant éventuellement de l'itération. Un pas de progression $s^{(k)}$ est alors calculé en résolvant le problème

$$\begin{aligned}
\text{minimiser} \quad & m^{(k)}(x^{(k)} + s) \\
\text{s.c.} \quad & \|s\|_k \leq \Delta^{(k)}
\end{aligned}$$

ou, à tout le moins, en assurant une réduction suffisante de l'approximation locale tout en satisfaisant la contrainte.

La fonction objectif $f(\tilde{x}^{(k)})$ est calculée au *point de test*

$$\tilde{x}^{(k)} = x^{(k)} + s^{(k)} \tag{2.27}$$

et comparée à la valeur prédite par l'approximation locale $m^{(k)}(\tilde{x}^{(k)})$. Si une réduction suffisante de la fonction objectif est obtenue, le point-test est accepté comme itéré suivant et le rayon de confiance est augmenté ou maintenu constant. Dans le cas contraire, le point-test est rejeté et la région de confiance est contractée, dans l'espoir de voir l'approximation locale donner de meilleures prédictions sur une région plus petite [20]. Formellement l'algorithme peut s'écrire

[3]La littérature spécialisée parle plus volontiers d'un *modèle* $m^{(k)}(x)$.

Algorithme 2.1 *Soit un point de départ* $x^{(0)}$*, un rayon de confiance initial* $\Delta^{(0)}$ *et les constantes* η_1*,* η_2*,* γ_1*,* γ_2 *qui satisfont aux conditions*

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{et} \quad 0 < \gamma_1 \leq \gamma_2 < 1. \tag{2.28}$$

Calculer $f(x^{(0)})$ *et initialiser* $k = 0$*.*

Étape 1 : Définition de l'approximation locale. *Choisir la norme* $\|\cdot\|_k$ *et définir une approximation locale* $m^{(k)}$ *dans* $\mathcal{B}^{(k)}$*.*

Étape 2 : Calcul d'un pas de progression. *Calculer un pas* $s^{(k)}$ *réduisant suffisamment l'approximation locale* $m^{(k)}$ *et tel que* $\tilde{x}^{(k)} = x^{(k)} + s^{(k)} \in \mathcal{B}^{(k)}$*.*

Étape 3 : Acceptation ou rejet du point-test. *Évaluer* $f(\tilde{x}^{(k)})$ *et définir le rapport*

$$\rho^{(k)} = \frac{f(x^{(k)}) - f(\tilde{x}^{(k)})}{m^{(k)}(x^{(k)}) - m^{(k)}(\tilde{x}^{(k)})}. \tag{2.29}$$

Si $\rho^{(k)} \geq \eta_1$*, définir* $x^{(k+1)} = \tilde{x}^{(k)}$ *; dans le cas contraire,* $x^{(k+1)} = x^{(k)}$*.*

Étape 4 : Mise à jour du rayon de confiance. *Choisir*

$$\Delta^{(k+1)} \in \begin{cases} [\Delta^{(k)}, +\infty[& \text{si } \rho^{(k)} \geq \eta_2, \\ [\gamma_2\Delta^{(k)}, \Delta^{(k)}] & \text{si } \rho^{(k)} \in [\eta_1, \eta_2[, \\ [\gamma_1\Delta^{(k)}, \gamma_2\Delta^{(k)}] & \text{si } \rho^{(k)} < \eta_1. \end{cases} \tag{2.30}$$

Augmenter ensuite k d'une unité et retourner à l'étape 1.

Les itérations pour lesquelles $\rho^{(k)} \geq \eta_1$ sont appelées des *itérations réussies* (« succesful iterations »), et nous notons l'ensemble de leurs indices par le symbole $\mathcal{S}$, *i.e.*

$$\mathcal{S} = \left\{ k \geq 0 \mid \rho^{(k)} \geq \eta_1 \right\}. \tag{2.31}$$

À l'opposé, nous définissons l'ensemble des *itérations infructueuses* (« unsuccessful iterations »)

$$\mathcal{U} = \left\{ k \geq 0 \mid \rho^{(k)} < \eta_1 \right\}. \tag{2.32}$$

De la même manière, nous posons

$$\mathcal{V} = \left\{ k \geq 0 \mid \rho^{(k)} \geq \eta_2 \right\} \tag{2.33}$$

l'ensemble des *itérations très réussies* (« very successful iterations »). Notons que $\mathcal{V} \subseteq \mathcal{S}$.

Cet algorithme de base laisse, pour l'instant, quelques zones d'ombre : le choix de l'approximation locale $m^{(k)}$, de la norme $\|\cdot\|_k$, la méthode employée pour calculer $s^{(k)}$ ainsi que la signification exacte de la périphrase « réduisant suffisamment l'approximation locale $m^{(k)}$ » et, enfin, la mise à jour pratique du rayon de

confiance. On peut remarquer que l'algorithme tel que décrit ci-dessus ne comporte pas de critère d'arrêt ; nous supposons donc qu'une suite infinie d'itérés $\{x^{(k)}\}$ est générée[4].

2.3 Globalisation par point proximal.

Nous devons également mentionner qu'il existe d'autres techniques de globalisation à côté de la recherche linéaire et des régions de confiance. Parmi ces techniques, celles dites du *point proximal* sont très proches, dans leur esprit, des régions de confiance. Les méthodes dites proximales sont déjà présentes dans la thèse de Martinet [70] mais l'algorithme du point proximal trouve ses pleins fondements dans les travaux de Rockafellar [89].

Dans toute sa généralité, cet algorithme est développé pour rechercher un zéro d'un opérateur maximal monotone, un de ses nombreux cadres d'application étant l'optimisation *convexe*. Dans ce contexte, l'algorithme du point proximal est caractérisé par une itération de base de la forme

$$x^{(k)} = \arg\min_x \left[f(x) + \frac{1}{\mu^{(k)}} \|x - x^{(k-1)}\|^2 \right] + e^{(k)} \tag{2.34}$$

où $e^{(k)}$ est introduit pour prendre en charge, d'un point de vue théorique, les erreurs liées au calcul numérique approché. Le terme de distance est introduit en vue de *régulariser* la fonction convexe $f(x)$ et ainsi assurer l'existence et l'unicité du minimum $x^{(k)}$. La convergence de l'algorithme est assurée moyennant certaines hypothèses. Parmi celles-ci, on trouve cependant l'hypothèse de convexité qui affaiblit sérieusement le résultat.

Fin des années 1980, l'essor de la théorie de la convergence variationnelle a permis d'introduire, dans l'algorithme de base, la notion de *perturbation* : la fonction $f(x)$ était remplacée, à l'itération k, dans le problème (2.34), par une autre fonction $f^{(k)}(x)$, la suite $f^{(k)}$ devant converger vers f. Cette approche facilite grandement la prise en charge de contraintes via l'exploitation des fonctions de *pénalisation* (voir par exemple [63, 98]).

Parallèlement, d'autres auteurs se sont plutôt penchés sur la *métrique* exploitée dans l'algorithme, *i.e.* sur la distance utilisée dans (2.34). C'est ainsi qu'apparaissent, d'une part, des métriques variant d'itération en itération (voir [17, 18, 6, 88]) et, d'autre part, une métrique fixe non-linéaire basée sur une méthode dite *entropique* (voir [16, 26, 55]). Bien entendu, les couplages entres ces différentes approches ont initié de nouvelles recherches (voir [2, 7]).

[4]En pratique, un critère d'arrêt doit naturellement être spécifié. Celui-ci doit stoppe le programme aussitôt que l'itéré $x^{(k)}$ satisfait l'utilisateur. La plupart des programmes spécifient également un nombre maximal d'itérations.

Plus récemment, Cartis *et al.* [13, 14] ont développé une méthode appelée « adaptive cubic overestimation » (ACO). Celle-ci présente des caractéristiques semblables à la méthode du point proximal.

Algorithme 2.2 *Soit un point de départ* $x^{(0)}$*, un rayon de confiance initial* $\Delta^{(0)}$ *et les constantes* η_1, η_2, γ_1, γ_2 *qui satisfont aux conditions* (2.28). *Calculer* $f(x^{(0)})$ *et initialiser* $k = 0$.

Étape 1 : Calcul d'un pas de progression. *Calculer un pas* $s^{(k)}$ *réduisant suffisamment l'approximation locale*

$$m^{(k)}(x^{(k)}+s) = f(x^{(k)}) + s^T \nabla_x f(x^{(k)}) + \frac{1}{2} s^T H^{(k)} s + \frac{1}{3}\frac{1}{\mu^{(k)}} \|s\|^3$$

où $H^{(k)}$ *est une approximation de la matrice hessienne* $\nabla_{xx} f(x^{(k)})$.

Étape 2 : Acceptation ou rejet du point-test. *Évaluer* $f(\tilde{x}^{(k)})$ *avec*

$$\tilde{x}^{(k)} = x^{(k)} + s^{(k)}$$

et définir le rapport

$$\rho^{(k)} = \frac{f(x^{(k)}) - f(\tilde{x}^{(k)})}{m^{(k)}(x^{(k)}) - m^{(k)}(\tilde{x}^{(k)})}.$$

Si $\rho^{(k)} \geq \eta_1$*, définir* $x^{(k+1)} = \tilde{x}^{(k)}$ *; dans le cas contraire,* $x^{(k+1)} = x^{(k)}$.

Étape 3 : Mise à jour du paramètre. *Choisir*

$$\frac{1}{\mu^{(k+1)}} \in \begin{cases}]0\,, 1/\mu^{(k)}] & \text{si } \rho^{(k)} > \eta_2, \\ [1/\mu^{(k)}\,, \gamma_1/\mu^{(k)}] & \text{si } \rho^{(k)} \in [\eta_1\,, \eta_2], \\ [\gamma_1/\mu^{(k)}\,, \gamma_2/\mu^{(k)}] & \text{si } \rho^{(k)} < \eta_1. \end{cases}$$

Augmenter ensuite k d'une unité et retourner à l'étape 1.

Nous pouvons constater une certaine similarité entre les méthodes proximales et les régions de confiance : les premières pénalisent l'éloignement entre deux itérés successifs alors que les secondes confinent l'itéré $x^{(k)}$ autour de $x^{(k-1)}$ grâce à une contrainte (la région de confiance). Là où, d'une part, l'intensité de la pénalité est gouvernée par le paramètre $\mu^{(k)}$ qui s'adapte d'itération en itération, nous avons, d'autre part, un rayon de confiance $\Delta^{(k)}$ variant également au cours des itérations qui détermine la taille de la région de « confinement ». De petites valeurs de $\mu^{(k)}$ provoquent une pénalité forte dont les effets sont similaires à une région de confiance de faible rayon $\Delta^{(k)}$. La régularisation des méthodes proximales agit comme une pénalisation — *i.e.* une forme de contrainte faible — alors que les régions de confiances utilisent une contrainte forte, infranchissable. Au vu de sa parenté avec les régions de confiance, cette technique de globalisation ne sera plus évoquée par la suite.

2.4 Globalisation par méta-heuristiques.

Le terme *méta-heuristique* caractérise une approche générale plutôt qu'une méthode à part entière. Les méthodes méta-heuristiques ne nécessitent aucun calcul de dérivée, mais uniquement des évaluations de la fonction objectif en différents points. Elles n'utilisent pas véritablement d'approximations locales. Certaines de ces méthodes ont fait leur preuves dans le domaine de l'optimisation *combinatoire* et *discrète* mais peuvent également être utilisées pour des problèmes d'optimisation continus et ont l'avantage de pouvoir être utilisée pour la recherche d'un minimum *global* de la fonction objectif f. Il convient néanmoins de se garder d'un enthousiasme excessif, ces méthodes reposent sur des analogies avec des mécanismes présents dans la *nature* ou sur des considérations géométriques et ont une base théorique relativement mince et partielle. Leur vitesse de convergence laisse également à désirer et le nombre d'évaluations de la fonction objectif est énorme en comparaison des méthodes basées sur des approximation locales. Parmi les méthodes rencontrées dans la littérature, nous n'en citerons que deux : la méthode du *recuit simulé* (simulated annealing) et les *algorithmes génétiques*.

2.4.1 Recuit simulé.

La méthode du recuit simulé [83, 85] a déjà été utilisée, par exemple, pour des identifications paramétriques[5] [71, 104]. Son principe est relativement simple et se base sur une analogie entre la minimisation d'une fonction et le refroidissement d'un métal en fusion. Quand un métal en fusion est refroidi suffisamment lentement, il tend à se solidifier dans un état d'énergie minimum. C'est le même principe qui gouverne le recuit simulé : au début, presque tous les mouvements (*i.e.* n'importe quelle point dans le voisinage de l'itéré courant $x^{(k)}$) sont acceptés comme itéré suivant. Ceci permet d'explorer l'espace des solutions. Ensuite, graduellement, la température diminue et, avec elle, la tolérance de l'algorithme. Et, finalement, seuls les mouvements entraînant une décroissance de la fonction objectif sont acceptés.

Synthétiquement, l'algorithme le plus commun fonctionne comme suit. La méthode est itérative. À l'itération k, la solution courante est $x^{(k)}$ et la valeur de la fonction objectif $f(x^{(k)})$. Un point $\tilde{x}^{(k)}$ est alors choisi aléatoirement dans le voisinage de $x^{(k)}$: si $f(\tilde{x}^{(k)}) < f(x^{(k)})$ le mouvement est accepté ($x^{(k+1)} = \tilde{x}^{(k)}$) et on passe à l'itération suivante. Dans le cas contraire, le mouvement a une certaine probabilité $P(\tilde{x}^{(k)}, k)$ d'être accepté et une probabilité $1 - P(\tilde{x}^{(k)}, k)$ d'être rejeté, auquel cas $x^{(k+1)} = x^{(k)}$ et on passe à l'itération suivante. La probabilité $P(\tilde{x}^{(k)}, k)$

[5] Le concept d'*identification paramétrique* est développé au chapitre 5.

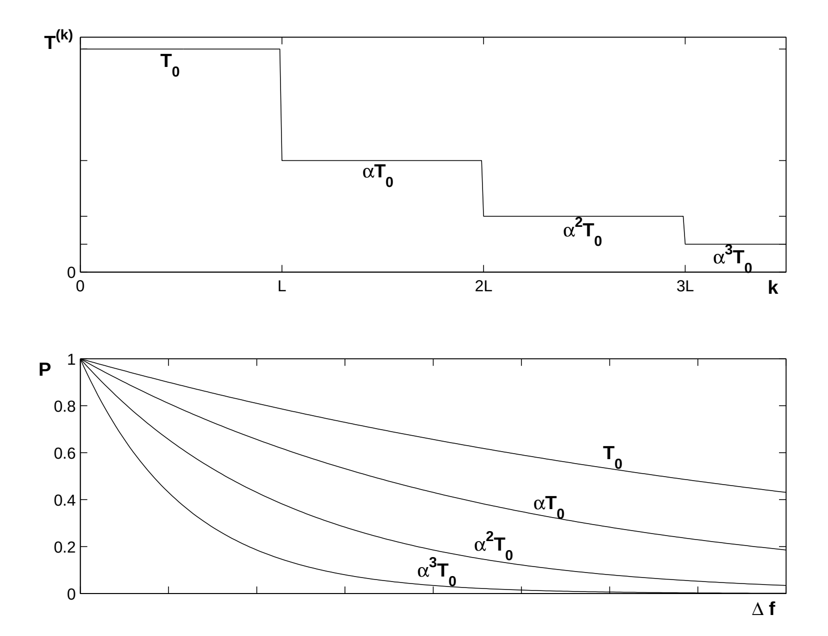


FIG. 2.1 – Lois de refroidissement de la méta-heuristique du recuit simulé. La figure du dessus représente l'évolution (2.36) de la température au fur et à mesure des itérations. La figure du dessous représente la probabilité d'acceptation (2.35) en fonction de la température et de la valeur de la différence $\Delta f = f(\tilde{x}^{(k)}) - f(x^{(k)})$.

d'acceptation de $\tilde{x}^{(k)}$ à l'itération k est calculée selon la formule suivante

$$P(\tilde{x}^{(k)},k) = \exp\left[-\frac{f(\tilde{x}^{(k)}) - f(x^{(k)})}{T^{(k)}}\right] \tag{2.35}$$

où $T^{(k)}$ est un paramètre nommé *température* décroissant avec les itérations. Le schéma usuel de décroissance de $T^{(k)}$ du type

$$T^{(k)} = \alpha^{k\,\mathrm{div}\,L}\, T_0 \tag{2.36}$$

où $0 < \alpha < 1$ est le *taux de refroidissement*, $T_0 > 0$ la *température initiale* et $L > 0$ un entier appelé *longueur des paliers*[6]. Ce sont les paramètres du schéma de refroidissement. Les lois de refroidissement (2.35) et (2.36) sont représentées sur la figure 2.1 Il convient bien entendu d'adjoindre à cet algorithme un critère d'arrêt.

2.4.2 Algorithmes génétiques.

Les méthodes génétiques [57, 83] sont également des méthodes ne nécessitant pas d'évalutation des dérivées de la fonction objectif. Elles se basent sur une sélection, puis une éventuelle amélioration, des meilleurs membres parmi un large échantillon de points, s'inspirant de la théorie de l'évolution de Darwin. Le vocabulaire utilisé est celui de l'étude des populations, on parle de *population* (ensemble de points) constituée d'*individus* (un point de cet ensemble) eux-mêmes caractérisés par des *gènes* (les valeurs des paramètres de cette solution).

Après avoir généré une population initiale, chaque itération (ou *génération*) se compose des étapes suivantes :

1. Évaluation de la fonction objectif pour chaque (nouvel) individu de la population.
2. Sélection des parents dans la population. Ceux-ci sont sélectionnés aléatoirement avec une plus forte probabilité pour les meilleurs individus (ceux dont la valeur de la fonction objectif est la plus faible).
3. Recombinaison des parents. Sélectionnés à l'étape précédente, ceux-ci sont croisés (avec un opérateur adéquat) afin de former de nouveaux membres de la population (les enfants) qui viennent s'ajouter aux précédents.
4. Amélioration locale des enfants. Ceux-ci sont éventuellement améliorés par une technique de recherche locale.

[6] L'opérateur div désigne la division entière.

5. Mutation de certains individus dans la population. Il s'agit de créer de nouveaux membres par de légères perturbations aléatoires des gènes de certains individus.
6. Survie des plus aptes au sein de la population. Certains individus sont sélectionnés aléatoirement dans la population et sont éliminés de celle-ci. La sélection aléatoire s'effectue suivant une probabilité qui favorise les meilleurs individus.

Il convient naturellement d'ajouter un critère d'arrêt à ce schéma, le plus simple étant d'arrêter le processus lorsque la différence entre les valeurs de fonction objectif entre le meilleur et le pire individu de la population est inférieur à un seuil fixé.

2.4.3 Propriétés générales.

Les méthodes méta-heuristiques présentent l'avantage de ne pas nécessiter de calcul des dérivées de la fonction objectif. Aucun coût d'implémentation ni de calcul du gradient de la fonction objectif n'est donc engendré. Ces méthodes ont également la propriété intéressante de pouvoir être dirigée vers le minimum global du problème et non vers un minimum local comme les méthodes d'ordre supérieur.

Toutefois, des applications numériques montrent que ces méthodes demandent un nombre d'évaluation de la fonction objectif très important [71]. Généralement le surcroît de temps de calcul engendré est prohibitif par rapport aux autres techniques de globalisation.

2.5 Conclusion.

Dans ce chapitre nous avons essentiellement présenté les deux techniques de globalisation les plus courantes : les méthodes avec recherche linéaire et les algorithmes avec régions de confiance. Les globalisations par méthode de type proximal et les méthodes méta-heuristiques n'y sont que succinctement évoquées dans un soucis de complétude.

L'étude systématique et complète de toutes les méthodes appartenant aux deux familles principales sort du cadre de ce travail. Certaines techniques d'optimisation unidimensionnelle sont présentées ici pour servir, d'une part, de source d'inspiration pour le développement d'autres algorithmes d'optimisation et, d'autre part, de points de comparaison. Le présent chapitre fait volontairement l'impasse sur les approximations locales nécessaires au calcul, respectivement, de la direction de recherche et du pas de progression au sein de la région de confiance. Le chapitre suivant est entièrement consacré à ces questions.

Chapitre 3

Approximations locales en optimisation mathématique

Ce chapitre traite du choix d'une approximation locale adéquate pour le calcul d'une direction de descente, dans le cas d'une globalisation par recherche linéaire, ou d'un pas de progression, dans le cas d'un algorithme avec régions de confiance. Les approximations locales développées dans ce chapitre peuvent être utilisées pour approcher la fonction objectif mais aussi, dans le cas de problèmes contraints, les contraintes. Les différentes combinaisons entre les techniques de globalisation et les approximations locales pour la fonction objectif et pour les contraintes constituent autant de méthodes différentes développées dans la littérature.

3.1 Approximations locales linéaires.

3.1.1 Méthode de la plus grande pente.

L'approximation locale la plus simple est sans conteste l'approximation linéaire

$$m^{(k)}(x^{(k)}+s) = f(x^{(k)}) + s^T g^{(k)} \tag{3.1}$$

où $g^{(k)}$ est le gradient $\nabla_x f(x^{(k)})$ de la fonction objectif au point $x^{(k)}$ (ou éventuellement une estimation de celui-ci).

Cette approximation locale donne naturellement naissance à la *méthode de la plus grande pente*. C'est une méthode avec recherche linéaire qui consiste à choisir, à chaque itération, la direction de descente $d^{(k)}$ qui présente la plus grande pente, c'est-à-dire l'opposé de l'estimation du vecteur $g^{(k)}$, *i.e.*

$$d^{(k)} = -g^{(k)}. \tag{3.2}$$

La méthode de la plus grande pente est globalement convergente et son ordre de convergence est linéaire. Cependant la vitesse de convergence s'avère très faible aussitôt que la fonction objectif présente une forme anisotrope, *i.e.* lorsque les valeurs propres du Hessien à l'optimum sont très différentes les unes des autres (voir par exemple [80]) .

3.1.2 Généralisation au cas non-différentiable.

Une généralisation de la méthode de la plus grande pente peut être obtenue pour des fonctions objectifs non-différentiables. Naturellement, le gradient de la fonction objectif n'est alors pas défini en tout point du domaine mais il est tout de même possible de définir et d'utiliser des généralisations appropriées.

Commençons par rappeler les concepts de base associés aux fonctions non-différentiables. Nous ne nous attarderons que sur le cas des fonctions continues, finies et convexes. Notons qu'une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est dite *convexe* si

$$f[\theta x + (1-\theta)y] \leq \theta f(x) + (1-\theta) f(y),\ \forall x,y \in \mathbb{R}^n, \forall \theta \in [0,1], \quad (3.3)$$

chaque fois que le second membre de cette inégalité est défini.

La dérivée unidirectionnelle, $f'_d(x)$, de f au point x dans la direction d est définie par

$$f'_d(x) = \lim_{t \to 0^+} \frac{f(x+td) - f(x)}{t}. \quad (3.4)$$

L'existence de cette dérivée est assurée pour une fonction f convexe et finie. Lorsque f est continûment différentiable dans un voisinage de x nous avons également $f'_d(x) = [\nabla_x f(x)]^T d$. Le *sous-différentiel* $\partial f(x)$ de f en x est l'ensemble

$$\partial f(x) = \{g \in \mathbb{R}^n : g^T d \leq f'_d(x), \forall d \in \mathbb{R}^n\} \quad (3.5)$$

et chaque élément du sous-différentiel de f en x est appelé un *sous-gradient* de f en x. Nous pouvons constater que si f est continûment dérivable dans un voisinage de x, le sous-différentiel de f en x se réduit à un singleton ne comprenant que le gradient $\nabla_x f(x)$. Le sous-différentiel peut également être écrit sous la forme

$$\partial f(x) = \{g \in \mathbb{R}^n : f(x) + g^T d \leq f(x+d), \forall d \in \mathbb{R}^n\}. \quad (3.6)$$

Une généralisation simple de la méthode de la plus grande pente est la *méthode du sous-gradient*. Il suffit de prendre pour direction de descente l'opposé d'un élément quelconque du sous-différentiel

$$g^{(k)} \in \partial f(x^{(k)}). \quad (3.7)$$

Mais cette technique s'avère plutôt inefficace en pratique et des exemples pour lesquels l'algorithme ne converge pas peuvent facilement être construits (voir par exemple [31]). Pour obtenir un algorithme globalement convergent, il faut pousser plus avant l'analogie avec la méthode de la plus grande pente et prendre $g^{(k)}$ comme l'opposé du sous-gradient de plus grande pente qui peut être caractérisée mathématiquement par

$$g^{(k)} = \arg \min_{g \in \partial f(x^{(k)})} \|g\|_2. \tag{3.8}$$

En effet, lorsque la dérivée existe, la direction (unitaire) de plus grande pente est en fait la solution du problème d'optimisation suivant

$$\begin{aligned} \arg \quad & \max_{d \in \mathbb{R}^n} |d^T g^{(k)}| && (3.9) \\ \text{s.c.} \quad & \|d\|_2 = 1, && (3.10) \\ & d^T g^{(k)} < 0. && (3.11) \end{aligned}$$

La condition (3.10) impose une valeur finie (unitaire) à la norme de la direction et la condition (3.11) restreint quant à elle l'espace de recherche aux directions de descente. Sachant que la pente est nécessairement négative, ce problème peut être reformulé plus simplement[1]

$$\begin{aligned} \arg \quad & \min_{d \in \mathbb{R}^n} d^T g^{(k)} && (3.12) \\ \text{s.c.} \quad & \|d\|_2 = 1, && (3.13) \end{aligned}$$

dont la solution $-g^{(k)}/\|g^{(k)}\|_2$ est bien un multiple de (3.2).

Par analogie, dans le cas non-différentiable, le sous-gradient de plus grande pente est la solution du problème d'optimisation

$$\begin{aligned} \arg \quad & \max_{g \in \partial f(x^{(k)})} \min_{d \in \mathbb{R}^n} d^T g && (3.14) \\ \text{s.c.} \quad & \|d\|_2 = 1. && (3.15) \end{aligned}$$

Le minimum de $d^T g$ soumis à la contrainte (3.15) s'obtient pour

$$d = -g/\|g\|_2, \tag{3.16}$$

quelle que soit g. La valeur du minimum est donc $-\|g\|_2$ et nous obtenons ainsi la formulation (3.8).

[1] Il suffit simplement d'utiliser l'équivalence entre les deux problèmes $\max f(x)$ et $\min[-f(x)]$.

3.2 Approximations locales quadratiques.

Pour améliorer l'adéquation entre la fonction objectif et l'approximation locale, il paraît naturel d'augmenter l'ordre de l'approximation et de passer à des approximations locales quadratiques de la forme

$$m^{(k)}(x^{(k)}+s) = f(x^{(k)}) + s^T g^{(k)} + \frac{1}{2} s^T H^{(k)} s. \tag{3.17}$$

Dans cette section nous considérerons, sauf indication contraire, que

$$g^{(k)} = \nabla_x f(x^{(k)}). \tag{3.18}$$

L'approximation est donc au moins du premier ordre.

3.2.1 Méthode de Newton.

L'approximation locale la plus immédiate est obtenue par un développement limité du second ordre de la fonction objectif

$$f(x^{(k)}+s) = f(x^{(k)}) + s^T g^{(k)} + \frac{1}{2} s^T \nabla_{xx} f(x^{(k)}) s + o(\|s\|^3) \tag{3.19}$$

et la matrice $H^{(k)}$ de l'approximation quadratique (3.17) est simplement le Hessien de la fonction objectif. Elle donne naissance à la *méthode de Newton*.

Dans la méthode de Newton *pure et simple*, le pas $s^{(k)}$ est calculé à chaque itération pour minimiser (3.17) et on pose $x^{(k+1)} = x^{(k)} + s^{(k)}$. Notons que $m^{(k)}$ ne possède un minimum unique que si $H^{(k)}$ est définie positive, la minimisation échouant dans les autres cas. Si $H^{(k)}$ est définie positive, $s^{(k)}$ est tel que

$$\nabla_x m^{(k)}(x^{(k)}+s^{(k)}) = 0, \text{ i.e. } H^{(k)} s^{(k)} = -g^{(k)}. \tag{3.20}$$

et nous constatons que la direction $s^{(k)}$ ainsi obtenue est une direction de descente car

$$g^{(k)T} s^{(k)} = -s^{(k)T} \nabla_{xx} f(x^{(k)}) s^{(k)} < 0 \tag{3.21}$$

lorsque la matrice hessienne est définie positive.

La méthode de Newton converge au second ordre, ce qui la rend très attractive. Elle présente cependant de périlleux désavantages. Tout d'abord, rien ne garantit que la matrice hessienne soit définie-positive pour toutes les itérations. Même s'il est possible de montrer sous de très légères hypothèses que la matrice hessienne est définie positive dans un voisinage d'un minimum local, encore faut-il que l'algorithme aboutisse dans ce voisinage. D'autre part, quand bien même la

matrice $H^{(k)}$ s'avérerait définie positive à chaque itération[2], la convergence globale ne serait pas garantie. En effet, si $s^{(k)}$ est bel et bien une direction de descente, rien ne permet d'affirmer que $f(x^{(k)}+s^{(k)}) < f(x^{(k)})$. Différents exemples de comportements problématiques peuvent être trouvés dans la littérature (voir par exemple [82]).

Le diagnostic à poser sur cette méthode est simple : aucune méthode de *globalisation* n'est utilisée. À aucun moment l'algorithme ne vérifie la décroissance des valeurs de la fonction objectif originelle $f(x^{(k)})$ aux itérés successifs. Une première variante, la *méthode de Newton avec recherche linéaire*, effectue donc une recherche linéaire dans la direction

$$d^{(k)} = -[H^{(k)}]^{-1} g^{(k)}. \tag{3.22}$$

Cette direction n'est toutefois une direction de descente que si $H^{(k)}$ est définie positive et la convergence globale de l'algorithme ne peut donc être établie.

Dans la seconde variante, la *méthode de Newton avec régions de confiance*, le pas de progression $s^{(k)}$ n'est pas calculé selon (3.20). La taille du pas de progression est limité par une région de confiance de rayon $\Delta^{(k)}$ et le pas de progression $s^{(k)}$ est donc la solution du problème

$$\arg \quad \min s^T g^{(k)} + \frac{1}{2} s^T H^{(k)} s \tag{3.23}$$

$$\text{s.c.} \quad \|s\| \leq \Delta^{(k)}. \tag{3.24}$$

Cette variante permet de circonvenir aux problèmes posés par des matrices hessiennes non définies positives. La résolution du sous-problème (3.23) dans le cas où la norme utilisée est la norme euclidienne a été étudiée par Moré et Sorensen [74] (voir section 6.1.1).

3.2.2 Méthodes de Newton modifiées.

De nombreuses variantes ont été proposées pour stabiliser la méthode de Newton. Elles passent généralement par une modification de la matrice $H^{(k)}$ destinée à assurer la convexité de l'approximation locale. Un Hessien modifié peut, par exemple, être obtenu en ajoutant simplement une matrice $E^{(k)}$ symétrique (semi) définie positive adéquatement choisie au véritable Hessien

$$H^{(k)} = \nabla_{xx} f(x^{(k)}) + E^{(k)}. \tag{3.25}$$

Le choix de $E^{(k)}$ est crucial pour les performances des algorithmes basés sur cette approximation. Intuitivement, il semble logique d'utiliser des matrices de correction $E^{(k)}$ aussi « petites » que possible. Sous certaines conditions, la convergence

[2]Ce qui est le cas, par exemple, pour une fonction objectif convexe.

globale de tels algorithmes avec une recherche linéaire peut être démontrée et leur taux de convergence est quadratique pour autant que le minimum ainsi atteint soit isolé (voir [80]). Supposons en effet que la suite d'itérés $\{x^{(k)}\}$ converge vers un point x^* pour lequel $\nabla_{xx} f(x^*)$ est suffisamment définie positive (la valeur propre la plus petite est supérieure à une certaine tolérance positive) pour qu'il ne soit plus nécessaire d'effectuer une correction pour toutes les itérations subséquentes. Les dernières itérations sont de pures itérations de Newton et le taux de convergence est dès lors identique. Dans les cas où $\nabla_{xx} f(x^*)$ est singulière ou presque singulière, il est possible que le taux de convergence soit diminué et que la convergence devienne linéaire.

L'idée la plus simple pour $E^{(k)}$ est certainement de trouver un scalaire $\nu^{(k)} > 0$ tel que la matrice

$$H^{(k)} = \nabla_{xx} f(x^{(k)}) + \nu^{(k)} I \tag{3.26}$$

soit définie positive. Ceci revient à imposer à la direction de Newton un biais croissant avec le paramètre $\nu^{(k)}$ vers la direction de plus grande pente. En effet, on constate que, si $\nu^{(k)}$ est extrêmement petit, la direction de descente obtenue est proche de la direction de Newton puisque

$$H^{(k)} \simeq \nabla_{xx} f(x^{(k)}). \tag{3.27}$$

Dans le cas contraire, si $\nu^{(k)}$ est extrêmement grand, le Hessien devient négligeable dans l'expression de $H^{(k)}$ et la direction de descente tend vers un multiple de la direction de plus grande pente

$$d^{(k)} \simeq -(\nu^{(k)} I)^{-1} g^{(k)} = -\frac{1}{\nu^{(k)}} g^{(k)}. \tag{3.28}$$

La correction (3.26) est à la base de la *méthode de Levenberg-Marquardt*. Cette façon de procéder peut également être interprétée comme une version affaiblie d'une région de confiance : l'approximation quadratique de Taylor (3.19) se voit ajouter un terme $\nu^{(k)} s^T s$ qui pénalise l'utilisation de grands pas de progression[3].

Il reste la question du choix de la valeur de $\nu^{(k)}$. Pour que la matrice (3.26) soit définie positive, il faut tout simplement que $\nu^{(k)}$ soit supérieur à la plus petite valeur propre de $\nabla_{xx} f(x^{(k)})$. Pour éviter le calcul explicite des valeurs propres, généralement coûteux, l'alternative suivante est parfois utilisée

$$\nu^{(k)} > \|\nabla_{xx} f(x^{(k)})\|_F \tag{3.29}$$

car la plus grande valeur propre (en valeur absolue) d'une matrice est bornée par

[3]La méthode de Levenberg-Marquardt peut aussi être interprétée comme une méthode proximale, voir à ce sujet la section 2.3.

la norme de Frobenius[4] de cette matrice. La valeur de $\nu^{(k)}$ générée par cette technique peut cependant s'avérer inutilement grande, biaisant ainsi fortement la direction de Newton vers la direction de plus grande pente. Les performances d'un algorithme basé sur ce type d'approximation peuvent dès lors en souffrir gravement. Il est possible de montrer que la modification (3.26) avec $\nu^{(k)} = -\lambda_{\min}$ est la modification $E^{(k)}$ qui possède la plus petite norme subordonnée à la norme euclidienne[5] et qui permette à $H^{(k)}$ d'être semi-définie positive.

En utilisant la norme de Frobenius pour $E^{(k)}$, nous obtenons la matrice de correction

$$E^{(k)} = Q^{(k)} \operatorname{diag}(\nu_i^{(k)})\, Q^{(k)T} \tag{3.31}$$

avec, pour $i = 1, \ldots, n$,

$$\nu_i^{(k)} = 0 \quad \text{si } \lambda_i > 0, \tag{3.32}$$

$$\nu_i^{(k)} > -\lambda_i \quad \text{sinon}, \tag{3.33}$$

où les λ_i et les colonnes de $Q^{(k)}$ sont respectivement les valeurs propres et les vecteurs propres de $\nabla_{xx} f(x^{(k)})$. En utilisant une tolérance δ (généralement la racine carrée de la précision machine), nous obtenons une troncature de la décomposition spectrale du Hessien

$$\nabla_{xx} f(x^{(k)}) = Q^{(k)} \operatorname{diag}(\lambda_i)\, Q^{(k)T} \tag{3.34}$$

qui est approché par

$$H^{(k)} = Q^{(k)} \operatorname{diag}\left[\max(\lambda_i, \delta)\right] Q^{(k)T}. \tag{3.35}$$

Cette approximation du Hessien est définie positive si $\delta > 0$ et semi-définie positive si $\delta \geq 0$. Cette technique peut cependant présenter des problèmes numériques comme l'illustrent Nocedal et Wright [80] par le problème suivant.

Exemple 3.1 *Considérons* $\nabla_x f(x^{(k)}) = (1, -3, 2)^T$ *et*

$$\nabla_{xx} f(x^{(k)}) = \operatorname{diag}(10, 3, -1) \tag{3.36}$$

[4]La norme de Frobenius de la matrice $A \in \mathbb{R}^n_m$ se définit comme

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2}. \tag{3.30}$$

[5]La norme euclidienne de la matrice $A \in \mathbb{R}^n_m$ est égale à la racine carrée de la plus grande des valeurs propres de la matrice $A^T A$.

qui n'est clairement pas définie positive et la direction de Newton (3.22) *n'est donc pas une direction de descente. En utilisant* $\delta = 10^{-8}$, *nous obtenons l'approximation du Hessien*

$$H^{(k)} = \text{diag}(10, 3, 10^{-8}) \tag{3.37}$$

qui est définie positive et dont les courbures dans les directions propres e_1, e_2 *ont été conservées. Avec une globalisation par recherche linéaire, la direction de descente qui en résulte est*

$$\begin{aligned} d^{(k)} &= -\left[H^{(k)}\right]^{-1} \nabla_x f(x^{(k)}) \\ &= -\frac{1}{10} e_1 - \frac{-3}{3} e_2 - \frac{2}{10^{-8}} e_3 \\ &\simeq -(2 \times 10^8) e_3. \end{aligned}$$

Ce pas de progression est presque parallèle à e_3 *et très grand. Malgré le fait qu'il s'agit bien d'une direction de descente, l'amplitude de ce pas de progression trahit un peu l'esprit de la méthode de Newton qui se base sur une approximation quadratique de la fonction objectif valide dans un voisinage de l'itéré* $x^{(k)}$. *Dès lors, la question de l'efficacité de cette direction de descente sur le calcul reste ouverte. Notons que si nous prenons* $\delta = 0$, *l'approximation locale ne présente plus de courbure du tout dans les directions de courbure négative. Malheureusement, la matrice* $H^{(k)}$ *est alors singulière (et semi-définie positive) et le calcul de la direction de descente doit se faire différemment.*

D'autres techniques de modification du Hessien ont été élaborées, certains auteurs proposent de simplement changer le signe des valeurs propres négatives dans la décomposition spectrale. Bien que pragmatique, cette stratégie paraît discutable. Afin d'épargner le calcul de la décomposition, d'autres auteurs effectuent une *décomposition de Cholesky modifiée* qui factorise le Hessien en modifiant certains éléments en cours de calcul pour assurer la définie positivité de la factorisation. Le lecteur intéressé est invité à consulter, par exemple, Nocedal et Wright [80].

3.2.3 Méthodes de type quasi-Newton.

Les méthodes de Newton (modifiées) ont un taux de convergence quadratique pourvu que $H^{(k)}$ demeure symétrique définie positive. Ces méthodes sont donc très attractives si ce n'est qu'elles nécessitent l'évaluation de dérivées secondes. Cet inconvénient peut s'avérer très gênant en pratique. La question cruciale de l'estimation des courbures s'est donc rapidement posée afin de pouvoir construire

des approximations du Hessien à chaque itération. Les méthodes de type *quasi-Newton* sont de celles-là : elles construisent des approximations successives $H^{(k)}$ de la matrice hessienne à partir du comportement de la fonction objectif et de son gradient au cours des itérations.

Pour ne pas recalculer $H^{(k)}$ à chaque itération, les méthodes de type quasi-Newton la mettent à jour d'une façon simple en prenant en compte les informations sur la courbure acquises depuis l'itération précédente. Supposons l'étape k d'un processus itératif : nous venons de générer un nouvel itéré $x^{(k)}$ et souhaitons construire une approximation locale quadratique $m^{(k)}$ de la forme (3.17). Quelles conditions devons nous imposer à $H^{(k)}$? Une condition raisonnable est d'imposer l'égalité des gradients de l'approximation

$$\nabla_x m^{(k)}(x^{(k)}+s) = g^{(k)} + H^{(k)}s \tag{3.38}$$

au point $x^{(k)}$ et en un autre point $\hat{x}^{(k-1)}$ pour lequel le gradient $\hat{g}^{(k-1)}$ est connu[6]. La première condition est automatiquement satisfaite : il suffit de prendre $s = 0$ dans (3.38) pour s'en convaincre. Nous obtenons de la seconde condition

$$\nabla_x m^{(k)}(\hat{x}^{(k-1)}) = g^{(k)} + H^{(k)}(\hat{x}^{(k-1)} - x^{(k)}) = \hat{g}^{(k-1)} \tag{3.39}$$

qui donne l'*équation sécante*

$$y^{(k)} = H^{(k)}r^{(k)} \tag{3.40}$$

en définissant les vecteurs

$$r^{(k)} = x^{(k)} - \hat{x}^{(k-1)}, \tag{3.41}$$

$$y^{(k)} = g^{(k)} - \hat{g}^{(k-1)}. \tag{3.42}$$

Le parallèle avec la méthode de Newton pure et simple nous donne un processus itératif

$$x^{(k+1)} = x^{(k)} - S^{(k)}g^{(k)} \tag{3.43}$$

où $S^{(k)} = (H^{(k)})^{-1}$ est une approximation de l'inverse de la matrice hessienne. Bien entendu, ce processus souffre des mêmes désavantages que la méthode de Newton pure et simple. Pour qu'elle soit efficace, il faut la coupler avec une technique de globalisation. Avec une globalisation par recherche linéaire, la direction de descente est tout naturellement

$$d^{(k)} = -S^{(k)}g^{(k)}. \tag{3.44}$$

[6]Habituellement, cet autre point est simplement l'itéré précédent $x^{(k-1)}$ si celui-ci est différent de $x^{(k)}$.

Dans ce cadre, il s'avère généralement plus avantageux de mettre à jour $S^{(k)}$ plutôt que $H^{(k)}$ et l'équation sécante (3.40) devient donc

$$r^{(k)} = S^{(k)} y^{(k)}. \tag{3.45}$$

Pour un certain écart $r^{(k)}$ et un changement de gradient $y^{(k)}$, l'équation sécante (3.40) (resp. (3.45)) fournit n conditions qui n'annihilent pas les $n(n+1)/2$ degrés de liberté de la matrice symétrique $H^{(k)}$ (resp. $S^{(k)}$). Il existe donc une infinité de méthodes de type quasi-Newton. Généralement, l'algorithme part d'une estimation initiale $H^{(0)}$ (souvent la matrice identité à défaut d'autre chose) et effectue une *mise à jour* de celle-ci au fur et à mesure des itérations en lui ajoutant une correction.

La plus simple des mises à jour ajoute une matrice *symétrique de rang un (SR1)*. On peut montrer que la seule mise à jour SR1 pour l'approximation du Hessien qui réponde à l'équation sécante (3.40) est

$$H_{\text{SR1}}^{(k)} = H^{(k-1)} + \frac{(y^{(k)} - H^{(k-1)} r^{(k)})(y^{(k)} - H^{(k-1)} r^{(k)})^T}{(y^{(k)} - H^{(k-1)} r^{(k)})^T r^{(k)}} \tag{3.46}$$

et que la seule mise à jour pour SR1 pour l'approximation de l'inverse du Hessien qui réponde à l'équation sécante (3.45) est

$$S_{\text{SR1}}^{(k)} = S^{(k-1)} + \frac{(r^{(k)} - S^{(k-1)} y^{(k)})(r^{(k)} - S^{(k-1)} y^{(k)})^T}{(r^{(k)} - S^{(k-1)} y^{(k)})^T y^{(k)}}. \tag{3.47}$$

Un des défauts majeurs de l'actualisation SR1 est que le dénominateur peut être nul. En fait, même si la fonction objectif est convexe et quadratique, il peut arriver qu'il n'y ait aucune mise à jour de rang un qui satisfasse l'équation sécante. Un autre défaut majeur est la non conservation de la définie positivité des matrices générées, c'est pourquoi la méthode SR1 n'est que rarement utilisée avec une globalisation par recherche linéaire. Cependant, le développement de la globalisation par régions de confiance, qui est capable de traiter des matrices non-définie positive, donne un second souffle à cette méthode d'actualisation. Pour des fonctions linéaires tout à fait générales, la mise à jour SR1 génère d'excellentes approximations du Hessien sous certaines conditions : ceci fait l'objet du théorème suivant [19].

Théorème 3.1 *Supposons f deux fois continûment différentiable et le Hessien borné et continu au sens de Lipschitz dans le voisinage d'un point $x^* \in \mathbb{R}^n$. Soit $\{x^{(k)}\}$ une suite d'itérés telle que $x^{(k)} \to x^*$. Supposons également que les écarts*

$r^{(k)} = x^{(k)} - \hat{x}^{(k-1)}$ *soient uniformément linéairement indépendants*[7]. *Alors les matrices* $H^{(k)}$ *générées par la formule d'actualisation SR1 satisfont à*

$$\lim_{k\to\infty} \|H^{(k)} - \nabla_{xx} f(x^*)\| = 0. \tag{3.49}$$

Des formules plus flexibles sont obtenues en effectuant des corrections de rang deux, *i.e.* pouvant être écrites sous la forme

$$H^{(k)} = H^{(k-1)} + uu^T + vv^T \tag{3.50}$$

avec $u, v \in \mathbb{R}^n$. Davidon [21] et Fletcher et Powell [34] ont proposé la loi d'actualisation suivante

$$H^{(k)}_{\mathrm{DFP}} = \left(I - \frac{y^{(k)} r^{(k)T}}{y^{(k)T} r^{(k)}}\right) H^{(k-1)} \left(I - \frac{r^{(k)} y^{(k)T}}{y^{(k)T} r^{(k)}}\right) + \frac{y^{(k)} y^{(k)T}}{y^{(k)T} r^{(k)}} \tag{3.51}$$

qui est connue sous l'acronyme *DFP*. En inversant cette matrice, nous obtenons une formule de mise à jour pour l'approximation de l'inverse de la matrice hessienne

$$S^{(k)}_{\mathrm{DFP}} = S^{(k-1)} + \frac{(r^{(k)} - S^{(k-1)} y^{(k)})(r^{(k)} - S^{(k-1)} y^{(k)})^T}{y^{(k)T}(r^{(k)} - S^{(k-1)} y^{(k)})}. \tag{3.52}$$

On peut montrer que $H^{(k)}_{\mathrm{DFP}}$ est la matrice symétrique satisfaisant l'équation sécante (3.40) la plus proche de $H^{(k-1)}$ (au sens d'une certaine norme pondérée de Frobenius), *i.e.* la solution du problème

$$\begin{aligned} \arg \quad & \min_H \left\| W^{1/2} \left(H - H^{(k-1)}\right) W^{1/2} \right\|_F \\ \text{s.c.} \quad & H = H^T, \quad H r^{(k)} = y^{(k)} \end{aligned} \tag{3.53}$$

où la matrice de pondération W peut être n'importe quelle matrice satisfaisant la relation $W y^{(k-1)} = r^{(k)}$. L'inverse de la *matrice hessienne moyenne* de la fonction objectif entre $\hat{x}^{(k-1)}$ et $x^{(k)}$

$$W = \left[\int_0^1 \nabla_{xx} f(\tilde{x}^{(k-1)} + \tau r^{(k)})\, d\tau\right]^{-1} \tag{3.54}$$

[7] Soit une suite de vecteurs finis $\{p^{(k)}\} \subset \mathbb{R}^n$, $p^{(k)} \neq 0$. Les vecteurs $p^{(k)}$ sont uniformément linéairement indépendants s'il existe une constante $\gamma > 0$ et des indices fixés $k_0 \geq 0$, $m \geq n$ tels que pour chaque $k \geq k_0$ et

$$\max_{j=k+1}^{k+m} \left\{ \frac{z^T p^{(j)}}{\|z\|\, \|p^{(j)}\|} \right\} \geq \gamma, \tag{3.48}$$

pour tout $z \in \mathbb{R}^n$, $z \neq 0$.

satisfait, par exemple, à cette relation.

La mise à jour DFP, bien que relativement efficace, a rapidement été dépassée par une autre formule de rang deux développée indépendamment par Broyden [8], Fletcher [28], Goldfarb [43] et Shanno [92] qui ont remplacé l'approximation du Hessien $H^{(k)}$ par son inverse $S^{(k)}$ dans la logique de construction du problème (3.53), *i.e.*

$$\begin{aligned} \text{arg} \quad & \min_S \left\| W^{1/2} \left(S - S^{(k-1)} \right) W^{1/2} \right\|_F \\ \text{s.c.} \quad & S = S^T, \quad S y^{(k)} = r^{(k)}. \end{aligned} \tag{3.55}$$

La formule de mise à jour ainsi obtenue est désignée par l'acronyme *BFGS*

$$S^{(k)}_{\text{BFGS}} = \left(I - \frac{r^{(k)} y^{(k)T}}{r^{(k)T} y^{(k)}} \right) S^{(k-1)} \left(I - \frac{y^{(k)} r^{(k)T}}{r^{(k)T} y^{(k)}} \right) + \frac{r^{(k)} r^{(k)T}}{r^{(k)T} y^{(k)}} \tag{3.56}$$

dont nous pouvons prendre l'inverse pour obtenir une mise à jour pour la matrice hessienne elle-même

$$H^{(k)}_{\text{BFGS}} = H^{(k-1)} + \frac{y^{(k)} y^{(k)T}}{r^{(k)T} y^{(k)}} - \frac{(H^{(k-1)} r^{(k)})(H^{(k-1)} r^{(k)})^T}{r^{(k)T} H^{(k-1)} r^{(k)}}. \tag{3.57}$$

De nombreux auteurs s'accordent pour dire qu'il s'agit de la meilleure formule pour les problèmes de minimisation non-contrainte.

Des analyses détaillées des propriétés de la méthode BFGS et plus largement des méthodes de type quasi-Newton peuvent être trouvées dans [31, 80]. Quelques commentaires généraux peuvent néanmoins être faits à propos des méthodes DFP et BFGS. Sous certaines conditions, toutes deux ont des propriétés théoriques qui garantissent un taux de convergence superlinéaire et une convergence globale lorsqu'elles sont utilisées avec une méthode de recherche linéaire. Théoriquement, la convergence globale de la méthode DFP requiert une recherche linéaire exacte alors que les conditions de Wolfe (2.5) et (2.6) (ou la condition forte (2.7)) suffisent pour BFGS. Cependant, les deux méthodes peuvent échouer pour des fonctions non-linéaires générales. En particulier, il apparaît que l'algorithme peut converger vers un *point de selle* car les conditions de convergence forcent l'approximation du Hessien (ou de son inverse) à être définie positive même sur un point de selle. Seule la convergence vers un *point critique du premier ordre* peut dès lors être garantie (voir section 4.1).

3.2.4 Directions conjuguées.

La méthode des directions conjuguées est basée sur des approximations quadratiques. Il s'agit de construire des directions conjuguées par rapport à une matrice $A \in \mathbb{R}^n_n$ symétrique définie positive. Deux directions non nulles et distinctes

$d, e \in \mathbb{R}^n$ sont conjuguées par rapport à A lorsque

$$d^T Ae = 0. \tag{3.58}$$

Vu le caractère défini positif de la matrice A, on peut montrer que $k \leq n$ directions conjuguées deux à deux sont linéairement indépendantes.

La *méthode du gradient conjugué* est un cas particulier de directions conjuguées : les directions sont obtenues par orthogonalisation des vecteurs gradients. Elle fut d'abord développée pour résoudre le système linéaire

$$Ax = b \tag{3.59}$$

où $b \in \mathbb{R}^n$. C'est l'équivalence de ce problème avec la minimisation de la fonction objectif quadratique et convexe

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c \tag{3.60}$$

où $c \in \mathbb{R}$ qui permet de l'utiliser comme méthode d'optimisation à part entière. La méthode des gradients conjugués procède par itérations. Pour fixer les idées, nous évoquerons sa formulation avec une globalisation par recherche linéaire.

La première direction de descente utilisée est celle de plus grande pente (3.2)

$$d^{(0)} = -g^{(0)}. \tag{3.61}$$

Les directions suivantes $d^{(k)}$ sont calculées à partir des composantes du gradient $g^{(k)}$ qui sont conjuguées aux k directions précédents $d^{(0)}, \ldots, d^{(k-1)}$. La direction $d^{(k)}$ est construite comme une combinaison du gradient $g^{(k)}$ et des directions antérieures

$$d^{(k)} = -g^{(k)} + \sum_{i=0}^{k-1} \gamma_i^{(k)} d^{(i)} \tag{3.62}$$

où les coefficients $\gamma_i^{(k)}$ sont calculés de façon à assurer la relation d'orthogonalité (3.58)

$$\gamma_i^{(k)} = \frac{g^{(k)T} A d^{(i)}}{d^{(i)T} A d^{(i)}}. \tag{3.63}$$

Les coefficients de la combinaison linéaire sont tous nuls à l'exception de $\gamma_{k-1}^{(k)}$ que nous renommons $\beta^{(k-1)}$. La direction de descente s'écrit donc

$$d^{(k)} = -g^{(k)} + \beta^{(k-1)} d^{(k-1)}. \tag{3.64}$$

Une caractéristique particulièrement intéressante de la méthode provient du fait que le coefficient $\beta^{(k-1)}$ peut être calculé sans faire intervenir explicitement la matrice A

$$\beta^{(k-1)} = \frac{g^{(k)T} \left(g^{(k)} - g^{(k-1)}\right)}{d^{(k-1)T} \left(g^{(k)} - g^{(k-1)}\right)}, \tag{3.65}$$

ce qui s'avère particulièrement utile pour les problèmes de grande taille. On peut montrer que le problème de minimisation de l'approximation (3.17) est résolu en, au plus, n itérations par la méthode du gradient conjugué [30, 80].

La méthode du gradient conjugué telle que décrite ci-dessus peut être utilisée avec des fonctions objectifs non quadratiques mais nécessite quelques légères modifications. Le coefficient $\beta^{(k-1)}$ peut, par exemple, être calculé à partir d'autres expressions équivalentes à (3.65) lorsque la fonction est quadratique. Il est conseillé, après n itérations, de réinitialiser le processus avec une itération de plus grande pente. Un nouveau jeu de directions de descente conjuguées pourra ainsi être créé lors des n itérations suivantes. De plus, vu que la recherche linéaire n'est jamais résolue de façon parfaitement exacte, il peut s'avérer que la direction $d^{(k)}$ ne pointe plus en direction d'une réduction de la fonction objectif (c'est en fait une direction de montée). Dans ce cas, on réinitialisera de même le processus par une itération avec la méthode de la plus grande pente.

D'autres méthodes utilisant le concept des directions conjuguées ont été développées dont certaines ne demandent pas l'évaluation du gradient de la fonction objectif. Le lecteur intéressé est invité à consulter [30]. Les directions conjuguées peuvent également être utilisées avec une globalisation par régions de confiance (voir, par exemple, [20, 44]).

3.2.5 Résolution d'équations non-linéaires.

Le principe de ces méthodes est de rechercher le vecteur des variables d'optimisation x^* solution du système d'équations implicites et non-linéaires

$$c_i(x) = \hat{c}_i \qquad i = 1, \ldots, m \tag{3.66}$$

en minimisant la fonction objectif

$$f(x) = \frac{1}{2} \sum_{i=1}^{m} [c_i(x) - \hat{c}_i]^2, \tag{3.67}$$

c'est l'écart au sens des *moindres carrés*. Étant donnée la présence de nombreuses sources d'erreurs (de modélisation, numériques, expérimentales,...), l'existence d'une telle solution n'est pas avérée. Néanmoins, loin de l'optimum, cette inexistence de solution n'influencera que peu la méthode de résolution proposée qui tend à converger vers une solution approchée du système de base. Les méthodes spécifiquement construites pour résoudre ce genre de problème nécessitent le calcul de la matrice jacobienne

$$G(x) = [\nabla_x c_1(x) \; \nabla_x c_2(x) \; \ldots \; \nabla_x c_m(x)] \tag{3.68}$$

de dimension $n \times m$.

Pour la résolution d'un système d'équations non-linéaires, nous pouvons utiliser la méthode de Newton–Raphson. Dans le cas de la minimisation d'une fonction objectif de type moindres carrés, on parlera plutôt de *la méthode de Gauss–Newton*. Considérons le développement de Taylor limité au premier ordre de chacune des composantes du vecteur

$$c(x) = [c_1(x)\ c_2(x)\ \dots\ c_m(x)]^T \tag{3.69}$$

autour de l'itéré $x^{(k)}$

$$c(x) \simeq c(x^{(k)}) + G^{(k)T}\,[x - x^{(k)}] \tag{3.70}$$

où $G^{(k)} = G(x^{(k)})$. Une approximation locale quadratique de la fonction peut être construite en utilisant (3.70) dans l'expression de la fonction objectif (3.67). Cette approximation est de la forme (3.17) avec

$$g^{(k)} = G^{(k)}[c(x^{(k)}) - \hat{c}] \tag{3.71}$$

et

$$H^{(k)} = G^{(k)}G^{(k)T}. \tag{3.72}$$

Cette expression est à comparer avec le gradient en un point x

$$\nabla_x f(x) = G(x)\,[c(x) - \hat{c}] \tag{3.73}$$

et la matrice hessienne en un point x

$$\nabla_{xx} f(x) = G(x)G(x)^T + \sum_{i=1}^{m} [c_i(x) - \hat{c}_i]\,\nabla_{xx} c_i(x). \tag{3.74}$$

On constate qu'il y a bien coïncidence entre les gradients de la fonction objectif et de l'approximation locale tandis que le second terme de l'expression de la matrice hessienne (3.74) est négligé.

Notons que d'autres formes de mesure de l'écart peuvent être utilisées (voir section 5.2.2). De manière analogue, le gradient et la matrice hessienne de la fonction objectif générale $f(x) = \mathcal{F}\,(c(x), \hat{c})$ sont respectivement de la forme

$$\nabla_x f(x) = G(x)\,\nabla_c \mathcal{F}\,(c(x), \hat{c}) \tag{3.75}$$

et

$$\nabla_{xx} f(x) = G(x)\,\nabla_{cc} \mathcal{F}\,(c(x), \hat{c})\,G(x)^T + \sum_{i=1}^{m} \frac{\partial \mathcal{F}\,(c(x), \hat{c})}{\partial c_i}\,\nabla_{xx} c_i(x). \tag{3.76}$$

L'approximation locale quadratique s'obtient alors en prenant pour matrice hessienne

$$H^{(k)} = G^{(k)}\,\nabla_{cc}\mathcal{F}\,(c(x^{(k)}),\hat{c})\,G^{(k)T}. \tag{3.77}$$

La *méthode de Levenberg–Marquardt* est très proche de la méthode de Gauss–Newton et constitue une stabilisation de celle-ci. Le Hessien de l'approximation locale est simplement augmenté d'un terme diagonal

$$H^{(k)} = G^{(k)}\,\nabla_{cc}\mathcal{F}\,(c(x^{(k)}),\hat{c})\,G^{(k)T} + \lambda_{LM}^{(k)} I \tag{3.78}$$

soit, dans le cas d'un écart mesuré au sens des moindres carrés (3.67)

$$H^{(k)} = G^{(k)} G^{(k)T} + \lambda_{LM}^{(k)} I \tag{3.79}$$

Plus le paramètre $\lambda_{LM}^{(k)}$ est grand, plus la variation des variables est atténuée : le terme diagonal ajouté est une sorte de pénalisation des déplacements. La mise à jour de ce paramètre est généralement réalisée par une procédure simple de test de décroissance. Lors de chaque itération, on calcule une nouvelle valeur des variables de contrôle $x^{(k+1)}$ où la valeur de la fonction objectif est ensuite évaluée. Si celle-ci est inférieure à la valeur correspondant à l'itéré courant, l'itération est acceptée et on passe à la suivante. Dans le cas contraire, on recommence l'itération en augmentant la valeur du paramètre λ_{LM} jusqu'à obtenir une itération acceptable.

3.2.6 Approximations quadratiques séparables.

Pour des problèmes de grandes tailles, divers auteurs ont proposé des approximations locales simplifiées pour être *séparables*, ce qui permet d'éviter les problèmes liés au stockage de la matrice hessienne ou de son approximation. Fleury [35] puis Zhang et Fleury [103] proposent de ne retenir que les termes diagonaux du Hessien de la fonction objectif au point $x^{(k)}$

$$H^{(k)} = \mathrm{diag}\left(\frac{\partial^2 f(x^{(k)})}{\partial x_1^2}, \ldots, \frac{\partial^2 f(x^{(k)})}{\partial x_n^2}\right). \tag{3.80}$$

Cette approximation peut éventuellement être assortie d'un gardien garantissant la définie-positivité de la matrice $H^{(k)}$.

Duysinx *et al.* [25] puis Bruyneel *et al.* [10] proposent de construire les termes diagonaux du Hessien de l'approximation locale à partir d'informations obtenues lors des itérations précédentes, ce qui permet d'éviter le calcul des dérivées secondes. Cette approximation est obtenue en remplaçant les termes diagonaux du

Hessien par une différence finie des gradients en deux points connus $x^{(k)}$ et $\tilde{x}^{(k-1)}$

$$H^{(k)} = \mathrm{diag}\left(\frac{g_1^{(k)} - \tilde{g}_1^{(k-1)}}{x_1^{(k)} - \tilde{x}_1^{(k-1)}}, \ldots, \frac{g_n^{(k)} - \tilde{g}_n^{(k-1)}}{x_n^{(k)} - \tilde{x}_n^{(k-1)}}\right). \tag{3.81}$$

Cette approximation peut évidemment donner lieu à des approximations négatives des courbures ; il arrive donc qu'un gardien algorithmique soit mis en place afin de garantir la définie-positivité de la matrice.

3.3 Autres approximations locales.

Les approximations locales les plus directes sont celles qui se basent sur le développement en série de Taylor (3.19). D'autres types d'approximations ont été développés : nous allons en brosser un rapide tableau. La première d'entre elles est l'approximation conique de Davidon [22]. Les méthodes suivantes nous proviennent du domaine de l'optimisation des structures qui a été particulièrement fécond en la matière. Des approximations locales convexes particulièrement efficaces y ont été développées en s'inspirant de problèmes-types : treillis, portiques, forme, topologie, etc. Cette famille de méthodes est connue dans le domaine sous le nom de *méthodes d'approximations convexes*. Elles sont, pour la plupart, disponibles dans le logiciel BOSS/Quattro [87].

3.3.1 Approximation conique.

L'approximation conique proposée par Davidon [22] s'inspire de l'approximation quadratique (3.17) en utilisant une *mise à échelle colinéaire* en lieu et place de s, *i.e.*

$$\frac{s}{1 - s^T a^{(k)}} \tag{3.82}$$

avec $a^{(k)} \in \mathbb{R}^n$. L'*approximation conique* prend la forme

$$m^{(k)}(x^{(k)} + s) = f(x^{(k)}) + \frac{s^T g^{(k)}}{1 - s^T a^{(k)}} + \frac{1}{2}\frac{s^T A^{(k)} s}{(1 - s^T a^{(k)})^2} \tag{3.83}$$

où $g^{(k)} \in \mathbb{R}^n$ et $A^{(k)} \in \mathbb{R}^n_n$ est une matrice symétrique. Le gradient de cette approximation est

$$\nabla_x m^{(k)}(x^{(k)} + s) = \frac{1}{1 - s^T a^{(k)}}\left(I + \frac{a^{(k)} s^T}{1 - s^T a^{(k)}}\right)\left(g^{(k)} + \frac{A^{(k)} s}{1 - s^T a^{(k)}}\right) \tag{3.84}$$

dont on peut constater, en prenant $s = 0$, la coïncidence avec le gradient de la fonction objectif au point $x^{(k)}$.

Le gradient s'annule si le dernier facteur de (3.84) prend la valeur nulle, *i.e.*

$$(A^{(k)} - g^{(k)} a^{(k)T}) s^{(k)} = -g^{(k)}. \tag{3.85}$$

Notons que la similitude entre les approximations coniques et quadratiques permet d'utiliser ici certains outils d'analyse des méthodes de type quasi-Newton (voir [22, 30] pour plus de détails).

3.3.2 Asymptotes mobiles.

La méthode des *asymptotes mobiles* de Svanberg [96] en généralisant une méthode proposée par Fleury et Braibant [36] s'est développée dans le domaine de l'optimisation des structures car les approximations locales proposées convenaient particulièrement bien aux fonctions objectifs généralement utilisées. Leur popularité tient au fait que les approximations générées sont *séparables*, ce qui simplifie considérablement la résolution des sous-problèmes tout en permettant d'envisager des problèmes de très grande taille. L'approximation locale prend la forme générale

$$\begin{aligned} m^{(k)}(x^{(k)} + s) &= f(x^{(k)}) + \sum_{i=1}^{n} p_i^{(k)} \left(\frac{1}{U_i^{(k)} - s_i} - \frac{1}{U_i^{(k)}} \right) \\ &\quad + \sum_{i=1}^{n} q_i^{(k)} \left(\frac{1}{s_i - L_i^{(k)}} - \frac{1}{L_i^{(k)}} \right) \end{aligned} \tag{3.86}$$

où les paramètres $p_i^{(k)}$ et $q_i^{(k)}$ sont calculés en fonction du signe de la composante correspondante du gradient au point $x^{(k)}$

$$p_i^{(k)} = (U_i^{(k)})^2 \left[\rho^{(k)} \frac{U_i^{(k)} - L_i^{(k)}}{2} + \max(0, g_i^{(k)}) \right], \tag{3.87}$$

$$q_i^{(k)} = (L_i^{(k)})^2 \left[\rho^{(k)} \frac{U_i^{(k)} - L_i^{(k)}}{2} - \min(0, g_i^{(k)}) \right]. \tag{3.88}$$

On peut montrer que cette approximation locale n'est convexe que si les paramètres $p_i^{(k)}$ et $q_i^{(k)}$ sont positifs, ce qui revient à imposer aux deux *asymptotes mobiles* $L_i^{(k)}$ et $U_i^{(k)}$ la condition suivante

$$L_i^{(k)} < 0 < U_i^{(k)} \quad \text{pour } i = 1, \ldots, n \tag{3.89}$$

et au *paramètre de non-monotonicité*[8] $\rho^{(k)}$ d'être positif. On vérifie aisément que l'approximation locale est du premier ordre et $\nabla_x m^{(k)}(x^{(k)}) = g^{(k)}$. Des règles empiriques de mise à jour ont été proposées par Svanberg [96, 97] pour les différents paramètres $U_i^{(k)}$, $L_i^{(k)}$ et $\rho^{(k)}$. Bruyneel [9] propose quant à lui d'utiliser la valeur de la fonction objectif et du gradient à un autre point connu $\tilde{x}^{(k)}$ pour mettre ces paramètres à jour.

D'autres approximations locales ont été développées sur la même base. La méthode originale de Svanberg [96] n'utilisait que des approximations monotones $\rho^{(k)} = 0$. Avant lui, Fleury et Braibant [36] avaient ouvert la voie vers ce type de méthodes en proposant ConLin (*Convex Linearization*), un cas particulier d'asymptotes mobiles, utilisant $\rho^{(k)} = 0$, $L_i^{(k)} = 0$ et $U_i^{(k)} \to +\infty$. Svanberg propose également de faire coïncider les dérivées secondes de l'approximation locale (3.86) et de la fonction objectif si celle-ci est disponible. Cependant, $m^{(k)}$ est séparable et seule la diagonale du Hessien peut être utilisée ; l'approximation locale doit donc être telle que

$$\frac{\partial^2 m^{(k)}(x^{(k)})}{\partial x_i^2} = \frac{\partial^2 f(x^{(k)})}{\partial x_i^2} \tag{3.90}$$

pour $i = 1, \ldots, n$. Les paramètres de l'expression (3.86) s'expriment alors comme suit

$$p_i^{(k)} = \frac{(U_i^{(k)})^3}{U_i^{(k)} - L_i^{(k)}} \left[g_i^{(k)} + \frac{1}{2} L_i^{(k)} \frac{\partial^2 f(x^{(k)})}{\partial x_i^2} \right], \tag{3.91}$$

$$q_i^{(k)} = \frac{(L_i^{(k)})^3}{U_i^{(k)} - L_i^{(k)}} \left[-g_i^{(k)} + \frac{1}{2} U_i^{(k)} \frac{\partial^2 f(x^{(k)})}{\partial x_i^2} \right]. \tag{3.92}$$

(3.93)

Ils sont généralement assortis d'un gardien les empêchant de prendre une valeur négative qui rendrait alors l'approximation non-convexe. Bruyneel [9] propose quant à lui d'estimer les dérivées secondes par différences finies (3.81) en utilisant la valeur du gradient $\hat{g}^{(k)}$ en un point connu $\hat{x}^{(k)}$.

3.4 Conclusion.

Ce chapitre traite du choix d'une approximation locale adéquate en découplant celui-ci de la technique de globalisation adoptée. Les approximations linéaires et

[8] On peut vérifier que, si $\rho^{(k)} = 0$, l'approximation locale séparable $m^{(k)}$ devient monotone en toutes ses variables, d'où le nom de *paramètre de non-monotonicité*.

quadratiques et leurs nombreuses variantes sont parmi les plus répandues : elles sont intuitives, simples et permettent dès lors une analyse rigoureuse. De nombreux ouvrages traitent de ces méthodes en détail. En fonction des problèmes envisagés, d'autres approximations locales ont été développées et s'avèrent particulièrement efficaces dans un domaine déterminé. Ainsi les stratégies d'asymptotes mobiles se sont avérées très utiles dans le domaine de l'optimisation des structures.

De très nombreuses méthodes envisagées dans la littérature sont, finalement, une combinaison particulière entre les techniques de globalisation et les approximations locales tant pour la fonction objectif que pour d'éventuelles contraintes. Bien des méthodes adoptent d'ailleurs des approximations locales différentes pour la fonction objectif et pour les contraintes, voire même pour différents types de contraintes (bornes, linéaires, non-linéaires, d'égalité ou d'inégalité). Le passage en revue complet des différentes combinaisons et de leurs propriétés sort du cadre de ce travail, le lecteur intéressé est invité à consulter les différents ouvrages référencés tout au long de ce chapitre. Les différentes méthodes que nous développerons dans les chapitres suivants reposeront, quant à elles, sur les approximations locales de type quadratique.

Chapitre 4

Convergence globale des algorithmes d'optimisation par régions de confiance

Notre travail s'intéresse particulièrement à l'optimisation par régions de confiance : nous détaillons ici les principaux résultats théoriques de ce domaine. Dans un soucis de simplicité, ce chapitre n'envisage que les problèmes non-contraints ; les problèmes contraints seront abordés au chapitre 8.

La plupart des algorithmes d'optimisation par régions de confiance sont élaborés à partir de l'algorithme élémentaire 2.1 détaillé dans la section 2.2. Les propriétés de convergence globale de cet algorithme sont bien établies ; nous allons détailler les plus remarquables d'entre elles. La plupart des éléments de ce chapitre sont tirés de l'excellent ouvrage de Conn, Gould et Toint [20] : le lecteur intéressé y trouvera les détails et les démonstrations de tous les théorèmes énoncés.

4.1 Points critiques du premier et du second ordre.

Il est impossible de prouver, en toute généralité, la convergence globale d'un algorithme d'optimisation vers un minimum global (voir section 1.2). Tout au plus devons nous nous contenter de *points critiques*. Les points critiques vérifient certaines propriétés nécessaires (mais non suffisantes) d'un minimum local.

On parle de *point critique du premier ordre* x^* si ce point satisfait à la condition nécessaire d'optimalité du premier ordre

$$\nabla_x f(x^*) = 0, \tag{4.1}$$

et de *point critique du second ordre* si — outre la condition (4.1) — x^* répond à

la condition nécessaire d'optimalité du second ordre

$$\nabla_{xx} f(x^*) \text{ semi-définie positive.} \tag{4.2}$$

4.2 Convergence globale vers un point critique du premier ordre.

Un algorithme est dit « globalement convergent » si les itérés successifs convergent vers un minimum local de la fonction objectif quel que soit le point de départ $x^{(0)}$.

La convergence de l'algorithme de base 2.1 peut être établie sous certaines hypothèses : quelques-unes portent sur la structure du problème lui-même et d'autres sur l'algorithme envisagé.

4.2.1 Hypothèses sur le problème.

Rappelons que nous cherchons une solution locale pour le problème non-contraint

$$\text{minimiser } f(x) \text{ avec } x \in \mathbb{R}^n \tag{4.3}$$

que nous supposons répondre aux hypothèses suivantes.

Hypothèse 4.1 *La fonction objectif $f : \mathbb{R}^n \to \mathbb{R}$ est deux fois continûment dérivable sur $\mathbb{R}^n$.*

Hypothèse 4.2 *Il existe une borne inférieure*[1] κ_{lbf} *telle que*

$$f(x) \geq \kappa_{lbf} \quad \forall x \in \mathbb{R}^n. \tag{4.4}$$

Hypothèse 4.3 *Le Hessien de la fonction objectif est borné, i.e. il existe une constante positive*[2] κ_{ufh} *telle que*

$$\|\nabla_{xx} f(x)\| \leq \kappa_{ufh} \quad \forall x \in \mathbb{R}^n. \tag{4.5}$$

Nous pouvons supposer, sans perte de généralité, que $\kappa_{ufh} \geq 1$.

---

[1] « lbf » pour « Lower Bound on the objective Function ».

[2] « ufh » pour « Upper bound on the objective Function's Hessian ».

4.2.2 Hypothèses sur l'algorithme.

L'algorithme 2.1 en lui-même doit, lui aussi, répondre à certaines hypothèses. Celles-ci résultent pour la plupart de deux préoccupations essentielles : la première est que l'approximation locale représente au mieux la fonction objectif dans la région de confiance et la seconde est que le problème approché ainsi créé ait une solution.

4.2.2.1 Hypothèses sur l'approximation locale.

Le but est ici de simplifier la démarche autant que possible sans pour autant masquer les idées maîtresses. Nous supposerons que l'approximation locale $m^{(k)}$ choisie à l'itération k pour représenter la fonction objectif dans la région de confiance $\mathcal{B}^{(k)}$ est une bonne approximation, lisse et du premier ordre de la fonction objectif. En conséquence, il est nécessaire de faire les hypothèses suivantes.

Hypothèse 4.4 *Pour tout k, l'approximation locale $m^{(k)}$ est deux fois continûment dérivable sur $\mathcal{B}^{(k)}$.*

Hypothèse 4.5 *Au point courant $x^{(k)}$, les valeurs de l'approximation locale et de la fonction objectif coïncident, i.e.*

$$m^{(k)}(x^{(k)}) = f(x^{(k)}) \quad \forall k. \tag{4.6}$$

Hypothèse 4.6 *Le gradient de l'approximation locale en $x^{(k)}$ est égal au gradient de la fonction objectif, i.e.*

$$g^{(k)} \triangleq \nabla_x m^{(k)}(x^{(k)}) = \nabla_x f(x^{(k)}) \quad \forall k. \tag{4.7}$$

Hypothèse 4.7 *Le Hessien de l'approximation locale reste borné en tout point de la région de confiance, i.e. il existe une constante*[3] $\kappa_{umh} \geq 1$ *telle que*

$$\|\nabla_{xx} m^{(k)}(x^{(k)})\| \leq \kappa_{umh} - 1 \quad \forall x \in \mathcal{B}^{(k)}, \forall k. \tag{4.8}$$

4.2.2.2 Hypothèses sur la résolution du problème approché.

Il est maintenant nécessaire de spécifier, à la deuxième étape de l'algorithme 2.1, les hypothèses auxquelles doit répondre le pas $s^{(k)}$: il s'agit de définir ce qu'est une « réduction *suffisante* de l'approximation locale ». Quelques concepts permettent de quantifier celle-ci.

La stratégie la plus simple pour réduire l'approximation locale $m^{(k)}$ dans la région de confiance est d'examiner le comportement de celle-ci dans la direction

[3] « umh » pour « Upper bound on the Model's Hessian ».

de plus grande pente $-g^{(k)}$. En conséquence, nous définissons l'*arc de Cauchy* comme étant le segment partant, dans cette direction, de l'itéré $x^{(k)}$ jusqu'à la frontière de la région de confiance $\mathcal{B}^{(k)}$ (voir figure 4.1) et le *point de Cauchy* comme le minimum de $m^{(k)}$ sur cet arc. Formellement, l'arc de Cauchy $\mathcal{C}^{(k)}$ à l'itération k de l'algorithme 2.1 est défini par

$$\mathcal{C}^{(k)} = \left\{ x \in \mathcal{B}^{(k)} : x = x^{(k)} - t\, g^{(k)},\ t \in \mathbb{R}^{+} \right\}. \tag{4.9}$$

Notons que cet arc se réduit à un point lorsque $g^{(k)} = 0$. Le point de Cauchy $x_C^{(k)}$ est, quant à lui, formellement défini par

$$x_C^{(k)} = \arg \min_{x \in \mathcal{C}^{(k)}} m^{(k)}(x). \tag{4.10}$$

Ce point joue un rôle central en théorie. Cependant, une minimisation exacte de $m^{(k)}$ sur l'arc de Cauchy $\mathcal{C}^{(k)}$ peut s'avérer difficile, c'est pourquoi on introduit le *point de Cauchy approché*, obtenu en réduisant l'approximation locale par retour arrière (« backtracking ») jusqu'à une certaine décroissance souhaitée *a priori*.

Soient les points

$$x^{(k)}(j) \triangleq x^{(k)} - \kappa_{bck}^{j} \frac{\Delta^{(k)}}{\|g^{(k)}\|_k} g^{(k)} \tag{4.11}$$

où $j \in \mathbb{N}$ et $\kappa_{bck} \in\,]0\,,1[$ est une constante donnée[4]. Soit j_c le plus petit naturel tel que la condition

$$m^{(k)}(x^{(k)}(j)) \leq m^{(k)}(x^{(k)}) + \kappa_{ubs}\,(x^{(k)}(j) - x^{(k)})^T g^{(k)} \tag{4.12}$$

soit vérifiée pour une constante donnée[5] $\kappa_{ubs} \in\,]0\,,1/2[$. Le point de Cauchy approché $x_{AC}^{(k)}$ à l'itération k de l'algorithme 2.1 est

$$x_{AC}^{(k)} \triangleq x^{(k)}(j_c). \tag{4.13}$$

Remarquons que le point de Cauchy approché répond à un critère similaire à la condition d'Armijo (2.5) pour les recherches linéaires (voir figure 4.2).

[4] « bck » pour « BaCKtraking ».
[5] « ubs » pour « Upper Bound on the Slope ».

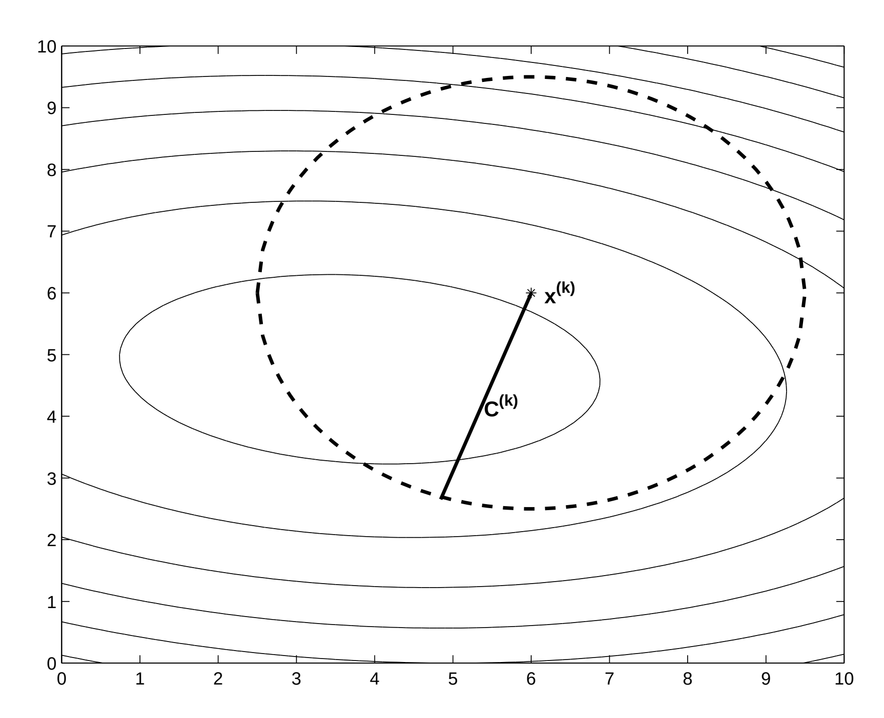


FIG. 4.1 – Région de confiance et arc de Cauchy. La frontière de la région de confiance est représentée par la ligne discontinue et l'arc de Cauchy par la ligne continue épaisse. Les fines courbes continues sont les lignes d'égales valeurs de l'approximation locale.

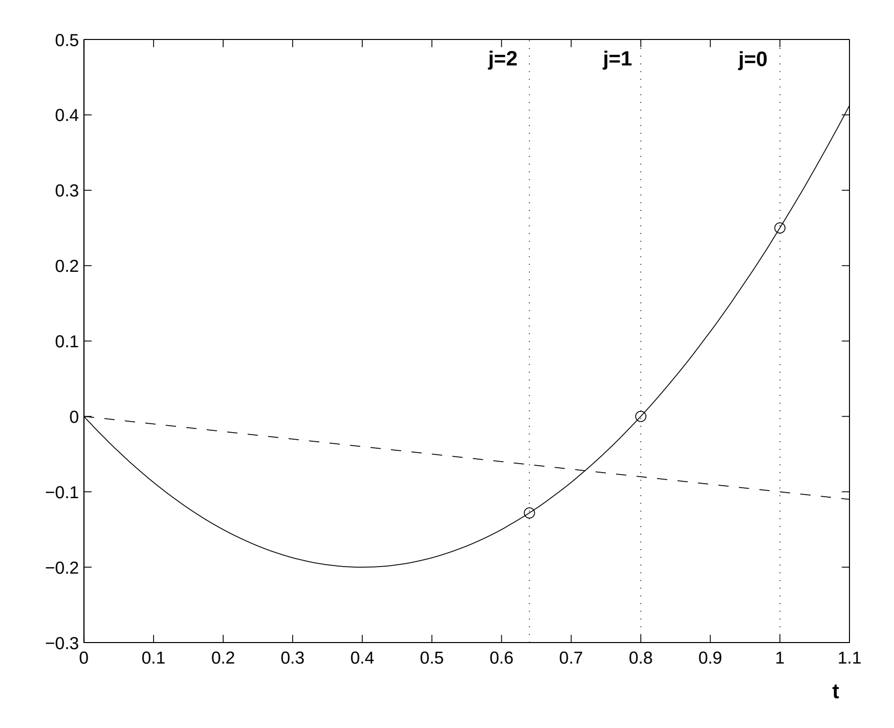


FIG. 4.2 – Point de Cauchy approché. L'approximation locale le long de la direction de plus grande pente $m^{(k)}(x^{(k)} - t\Delta^{(k)} g^{(k)}/\|g^{(k)}\|_k)$ est représenté par la courbe continue et le membre de droite $m^{(k)}(x^{(k)}) + t\kappa_{ubs}(x^{(k)}(j) - x^{(k)})^T g^{(k)}$ de (4.12) est représenté par la droite discontinue. Pour fixer les idées, les valeurs des constantes sont $\kappa_{ubs} = 0,1$ et $\kappa_{bck} = 0,8$. Sur l'exemple proposé, le point de Cauchy approché $x_{AC}^{(k)}$ correspond donc à $j_c = 2$.

En nous servant de ces définitions, nous avons le théorème suivant.

Théorème 4.1 *Si l'hypothèse 4.4 est satisfaite, le point de Cauchy approché* $x_{AC}^{(k)}$ *est bien défini dans le sens où* j_c *existe et est fini. De plus, la décroissance de l'approximation locale est minorée*

$$m^{(k)}(x^{(k)}) - m^{(k)}(x_{AC}^{(k)}) \geq \kappa_{dcp} \|g^{(k)}\| \min\left\{ \frac{\|g^{(k)}\|}{\beta^{(k)}}, \nu_C^{(k)} \Delta^{(k)} \right\} \tag{4.14}$$

où

$$\beta^{(k)} \triangleq 1 + \max_{x \in \mathcal{B}^{(k)}} \|\nabla_{xx} m^{(k)}(x)\|, \quad \nu_C^{(k)} = \frac{\|g^{(k)}\|}{\|g^{(k)}\|_k} \tag{4.15}$$

et $\kappa_{dcp} \in \,]0\,,1[$ *est une constante*[6] *indépendante de k.*

Nous ne ferons désormais plus la distinction entre point de Cauchy et point de Cauchy approché. Nous utiliserons donc la dénomination « point de Cauchy » et son symbole $x_C^{(k)}$ pour les deux formes exacte et approchée.

Nous pouvons constater que la décroissance de l'approximation locale au point de Cauchy dépend de la valeur de $\nu_C^{(k)}$, à tout le moins pour de petites valeurs du rayon de confiance $\Delta^{(k)}$. Nous verrons que l'hypothèse 4.9 nous permet d'affirmer que

$$\nu_C^{(k)} \geq \frac{1}{\kappa_{une}} > 0 \tag{4.16}$$

pour tout k. Il est donc acceptable d'exiger qu'à chaque itération l'approximation locale décroisse au moins d'une fraction donnée de celle obtenue au point de Cauchy.

Hypothèse 4.8 *Pour tout k, le pas de progression* $s^{(k)}$ *est tel que*

$$m^{(k)}(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_{mdc} \|g^{(k)}\| \min\left\{ \frac{\|g^{(k)}\|}{\beta^{(k)}}, \Delta^{(k)} \right\} \tag{4.17}$$

où $\kappa_{mdc} \in \,]0\,,1[$ *est une constante*[7].

Cette hypothèse a l'intéressante conséquence suivante.

Théorème 4.2 *Si les hypothèses 4.4 et 4.8 sont satisfaites et si*

$$\nabla_x f(x^{(k)}) \neq 0, \tag{4.18}$$

alors $m(x^{(k)} + s^{(k)}) < m^{(k)}(x^{(k)})$ *et* $s^{(k)} \neq 0$.

[6] « dcp » pour « Decrease at the Cauchy Point ».
[7] « mdc » pour « Model DeCrease ».

Dans ce cas, la décroissance de l'approximation locale est assurée pour autant que $x^{(k)}$ ne soit pas un point critique du premier ordre.

Naturellement, nous pouvons décider de faire décroître l'approximation locale en deçà de la borne donnée par l'hypothèse 4.8. En particulier, nous pouvons être amené à trouver un minimum exact de l'approximation locale dans la région de confiance

$$x_M^{(k)} = \arg \min_{x \in \mathcal{B}^{(k)}} m^{(k)}(x) \tag{4.19}$$

ou une approximation de celui-ci. En effet, le résultat suivant nous garantit la validité d'une telle approche.

Théorème 4.3 *Si, pour tout k, le pas de progression $s^{(k)}$ est tel que*

$$m^{(k)}(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_{amm} \left[m^{(k)}(x^{(k)}) - m^{(k)}(x_M^{(k)}) \right] \tag{4.20}$$

où $\kappa_{amm} \in]0\,,1]$ est une constante[8]*, l'hypothèse 4.8 est satisfaite pour une valeur constante κ_{mdc} choisie en conséquence.*

4.2.2.3 Hypothèse sur les régions de confiance.

Il reste à formuler une dernière hypothèse sur les différentes normes $\|\cdot\|_k$ utilisées pour définir les régions de confiance, celles-ci ne pouvant pas s'étendre ou se contracter asymptotiquement dans une direction au fur et à mesure des itérations.

Hypothèse 4.9 *Il existe une constante*[9] $\kappa_{une} \geq 1$ *telle que*

$$\frac{1}{\kappa_{une}} \|x\|_k \leq \|x\| \leq \kappa_{une} \|x\|_k \quad \forall x \in \mathbb{R}^n, \forall k. \tag{4.21}$$

On dit alors que la norme $\|\cdot\|_k$ est *uniformément équivalente* à la norme euclidienne.

4.2.3 Théorème de convergence.

Il est maintenant possible de prouver que l'algorithme 2.1 est globalement convergent vers un point critique du premier ordre. Plus précisément, tous les points limites x^* de suites $\{x^{(k)}\}$ générées par l'algorithme sont des points critiques du premier ordre pour le problème (4.3), c'est-à-dire qu'ils satisfont à

$$\nabla_x f(x^*) = 0, \tag{4.22}$$

[8] « amm » pour « Approximate Model Minimizer ».
[9] « une » pour « Uniform Norm Equivalence ».

indépendamment de la position du point de départ $x^{(0)}$ et du choix du rayon de confiance initial $\Delta^{(0)}$.

Le théorème suivant permet de l'affirmer.

Théorème 4.4 *Si les hypothèses 4.1–4.9 sont satisfaites, on a*

$$\lim_{k\to\infty} \|\nabla_x f(x^{(k)})\| = 0. \tag{4.23}$$

Ceci signifie que tout point limite d'une suite d'itérés est un point critique du premier ordre. La preuve complète de ce théorème peut être trouvée dans [20].

Notons qu'il est impossible d'assurer que la suite d'itérés converge vers un minimum global de f, ce résultat est en effet totalement illusoire en l'absence d'hypothèse supplémentaire sur la fonction objectif.

4.3 Convergence globale vers un point critique du second ordre.

Il va de soi que le mieux que nous puissions espérer si nous ne requérons de notre approximation locale que la coïncidence avec la fonction objectif et son gradient est que notre algorithme converge vers un point critique du premier ordre. Si une convergence plus forte est requise nous devrons évidemment exploiter l'information du second ordre.

4.3.1 Approximations locales asymptotiquement convexes.

La première étape est de déterminer sous quelles conditions nous pouvons assurer que, non seulement la suite $\{g^{(k)}\}$ converge vers zéro, mais aussi que la suite $\{x^{(k)}\}$ converge. Ceci dépend des termes du second ordre de l'approximation locale. On peut montrer que, si les approximations locales $m^{(k)}$ sont convexes tout au long d'une sous-suite d'itérations convergeant vers un point critique (isolé) du premier ordre — qui peut éventuellement être un point de selle — alors il y a convergence vers ce point de la suite complète même si la convexité de l'approximation locale ne reflète pas la véritable courbure de la fonction objectif.

En conséquence, nous ne devons pas seulement nous assurer que l'algorithme converge vers un minimum isolé de la fonction objectif mais encore que l'approximation locale et la fonction objectif coïncident au second ordre si les itérés s'approchent d'un point critique du premier ordre. Plus formellement, nous faisons donc l'hypothèse suivante.

Hypothèse 4.10 *Nous supposons que*

$$\lim_{k\to\infty} \left\| \nabla_{xx} f(x^{(k)}) - \nabla_{xx} m^{(k)}(x^{(k)}) \right\| = 0 \tag{4.24}$$

lorsque

$$\lim_{k\to\infty} \|\nabla_x f(x^{(k)})\| = 0. \tag{4.25}$$

Cette hypothèse nous préserve des situations où l'algorithme pourrait converger vers un point de selle. Notons que la satisfaction de cette dernière hypothèse et de l'hypothèse 4.1 entraîne la satisfaction de l'hypothèse 4.7 que nous pouvons donc ignorer aussi longtemps que nous adoptons l'hypothèse 4.10.

Nous pouvons analyser les conséquences de cette nouvelle hypothèse imposée à l'approximation locale sur la convergence de l'algorithme grâce au théorème suivant.

Théorème 4.5 *Supposons que les hypothèses 4.1–4.10 sont satisfaites, que $x^{(k_i)}$ est une sous-suite des itérés générés par l'algorithme 2.1 qui converge vers un point critique du premier ordre x^*, que $s^{(k)} \neq 0$ pour tout k suffisamment grand et que $\nabla_{xx} f(x^*)$ est définie positive. Dans ce cas la suite complète des itérés $\{x^{(k)}\}$ converge vers x^* et toutes les itérations finissent par être très réussies.*

Ce théorème prouve que, si un des points limites est un minimum isolé, alors l'algorithme converge vers ce minimum et le rayon de confiance $\Delta^{(k)}$ devient inutile dans le calcul du pas de progression. Ceci signifie que le taux de convergence de l'algorithme 2.1 est complètement déterminé par la méthode utilisée pour calculer le pas de progression $s^{(k)}$ lorsque la contrainte de confinement dans la région de confiance est inactive (*i.e.* lorsque $\|s^{(k)}\|_k < \Delta^{(k)}$). La preuve de ce résultat très important peut être trouvée dans [20].

4.3.2 Approximations locales non-convexes.

Nous souhaitons maintenant explorer les possibilités de convergence de la suite d'itérés vers un point critique du second ordre lorsque le point limite ne répond pas à la condition de définie positivité de $\nabla_{xx} f$. Naturellement, la matrice hessienne au point minimum doit tout de même être semi-définie positive[10]. Intuitivement, la convergence ne s'opère que si l'algorithme est capable de détecter et d'éviter un maximum ou un point de selle. Une manière d'opérer est de tirer avantage des directions de courbure négative quand elles existent.

[10] Il existe deux cas de points critiques du second ordre dont la matrice hessienne n'est *que* semi-définie positive : lorsque celui-ci est un minimum non-isolé ou un point d'inflexion multidimensionnel.

Nous supposerons que le Hessien de l'approximation locale au point courant

$$H^{(k)} = \nabla_{xx} m^{(k)}(x^{(k)}) \tag{4.26}$$

a au moins une valeur propre strictement négative $\tau^{(k)}$. Nous pouvons en conséquence déterminer une direction $u^{(k)}$ telle que

$$u^{(k)T} g^{(k)} \leq 0,\ \|u^{(k)}\|_k = \Delta^{(k)} \text{ et } u^{(k)T} H^{(k)} u^{(k)} \leq \kappa_{snc} \tau^{(k)} (\nu_E^{(k)} \Delta^{(k)})^2 \tag{4.27}$$

où la constante[11] $\kappa_{snc} \in]0,1]$ et

$$\nu_E^{(k)} = \frac{\|u^{(k)}\|}{\|u^{(k)}\|_k}. \tag{4.28}$$

Ceci signifie que $u^{(k)}$ est une direction de descente, de norme égale au rayon de confiance et ayant une composante significative dans la direction des vecteurs propres de $H^{(k)}$ correspondant à des valeurs propres négatives. Pratiquement on peut prendre pour direction $u^{(k)}$ une approximation du vecteur propre correspondant à la valeur propre $\tau^{(k)}$ dont le signe et la norme sont choisis afin de respecter les deux premières conditions de (4.27). Nous minimisons ensuite l'approximation locale dans cette direction tout en restant à l'intérieur de la région de confiance $\mathcal{B}^{(k)}$. Formellement, nous calculons un point analogue au point de Cauchy nommé *point propre* $x_E^{(k)}$ de l'itération k de l'algorithme 2.1 tel que

$$x_E^{(k)} = x^{(k)} + t_E^{(k)} u^{(k)} \in \mathcal{B}^{(k)} \tag{4.29}$$

et

$$m^{(k)}(x_E^{(k)}) = m^{(k)}(x^{(k)} + t_E^{(k)} u^{(k)}) = \min_{t \in]0,1]} m^{(k)}(x^{(k)} + t\, u^{(k)}). \tag{4.30}$$

Comme pour le point de Cauchy, une minimisation exacte de $m^{(k)}$ dans la région de confiance $\mathcal{B}^{(k)}$ peut s'avérer difficile. On introduit dès lors un *point propre approché* qui réduit la valeur de l'approximation locale par retour arrière (backtracking) jusqu'à une certaine décroissance souhaitée *a priori*.

Soient les points

$$x^{(k)}(j) \triangleq x^{(k)} - \kappa_{bck}^j u^{(k)} \tag{4.31}$$

où $j \in \mathbb{N}$ et $\kappa_{bck} \in]0\,,1[$ est une constante donnée. Soit j_e le plus petit naturel tel que la condition

$$m^{(k)}(x^{(k)}(j)) \leq m^{(k)} x^{(k)} + \kappa_{ubc} \tau^{(k)} \left(\kappa_{bck}^j \|u^{(k)}\| \right)^2 \tag{4.32}$$

[11] « snc » pour « Sufficient Negative Curvature ».

soit vérifiée pour une constante[12] $\kappa_{ubc} \in \left]0\,,\,\frac{1}{2}\kappa_{snc}\right[$ donnée. Le point propre approché $x_{AE}^{(k)}$ à l'itération k de l'algorithme 2.1 est alors défini par

$$x_{AE}^{(k)} \triangleq x^{(k)}(j_e). \tag{4.33}$$

Comme pour le point de Cauchy nous pouvons abandonner la distinction entre le point propre $x_E^{(k)}$ et le point propre approché $x_{AE}^{(k)}$ et retenir uniquement la notation et le nom du premier. En effet, il est possible de démontrer que ces deux points mènent à des réductions semblables de l'approximation locale (voir [20]).

4.3.3 Hypothèses sur l'algorithme.

Il est évident que la convergence globale de l'algorithme vers un point critique du second ordre, ne peut être obtenue qu'au prix d'hypothèses supplémentaires. Les deux hypothèses suivantes sur l'algorithme 2.1 paraissent raisonnables.

Hypothèse 4.11 *Les Hessiens $\nabla_{xx} m^{(k)}$ des approximations locales $m^{(k)}$ de l'algorithme 2.1 sont uniformément continus au sens de Lipschitz sur son domaine de définition, c'est-à-dire qu'il existe une constante*[13] $\kappa_{lch} > 0$ *telle que*

$$\|\nabla_{xx} m^{(k)}(x) - \nabla_{xx} m^{(k)}(y)\| \leq \kappa_{lch} \|x - y\| \tag{4.34}$$

pour tout $x, y \in \mathcal{B}^{(k)}$ et tout $k \in \mathbb{N}$.

Hypothèse 4.12 *Soit $\tau^{(k)}$ la plus petite valeur propre du Hessien de l'approximation locale au point courant $\nabla_{xx} m^{(k)}(x^{(k)})$. Si $\tau^{(k)} < 0$, alors*

$$m^{(k)}(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_{sod} \max\left\{ \|g^{(k)}\| \min\left[\frac{\|g^{(k)}\|}{\beta^{(k)}}, \Delta^{(k)}\right], |\tau^{(k)}| \min\left[(\tau^{(k)})^2, (\Delta^{(k)})^2\right] \right\}$$

pour une constante $\kappa_{sod} \in \left]0, \frac{1}{2}\right[$ donnée[14].

L'hypothèse 4.11 assure simplement que l'hypothèse 4.12 puisse être imposée. Cette dernière implique que, si une courbure négative apparaît dans l'approximation locale lorsqu'un point critique du premier ordre est approché, et si le point propre donne une réduction de l'approximation locale inférieure à celle obtenue au point de Cauchy, alors cette courbure négative doit être exploitée par la procédure de calcul du pas de progression $s^{(k)}$.

---

[12] « ubc » pour « Upper Bound on the Curvature ».
[13] « lch » pour « Lipschitz Constant for the model's Hessian »
[14] « sod » pour « Second Order Decrease »

Notons que l'hypothèse 4.12 est automatiquement satisfaite si le pas de progression $s^{(k)}$ est calculé en vue d'approcher $x_M^{(k)}$, le minimum exact (4.19) de l'approximation locale au sein de la région de confiance. Ceci signifie que, si $s^{(k)}$ est tel que la décroissance de l'approximation locale est au moins égale à une fraction déterminée de la décroissance obtenue au point $x_M^{(k)}$, *i.e.*

$$m^{(k)}(x^{(k)}) - m^{(k)}(x^{(k)} + s^{(k)}) \geq \kappa_{amm} \left[m^{(k)}(x^{(k)}) - m^{(k)}(x_M^{(k)}) \right], \tag{4.35}$$

alors l'hypothèse 4.12 est satisfaite. La démonstration de cette dernière affirmation peut s'effectuer en tenant compte de la relation évidente

$$m^{(k)}(x_M^{(k)}) \leq \min \left[m^{(k)}(x_E^{(k)}), m^{(k)}(x_C^{(k)}) \right]. \tag{4.36}$$

Une dernière hypothèse, pourtant peu contraignante, permet d'obtenir un théorème de convergence très intéressant. Il s'agit de modifier légèrement les conditions d'actualisation du rayon de confiance de sorte que celui-ci augmente bel et bien — mais pas démesurément — lors d'itérations très réussies.

Hypothèse 4.13 *Si* $\rho^{(k)} \geq \eta_2$ *et* $\Delta^{(k)} \leq \Delta_{\max}$ *alors*

$$\Delta^{(k+1)} \in [\gamma_3 \Delta^{(k)}, \gamma_4 \Delta^{(k)}] \tag{4.37}$$

pour des constantes données $\gamma_4 \geq \gamma_3 > 1$ *et* $\Delta_{\max} > 0$*. Par contre, si* $\rho^{(k)} \geq \eta_2$ *et* $\Delta^{(k)} > \Delta_{\max}$*, il convient de prendre* $\Delta^{(k+1)} \geq \Delta^{(k)}$*.*

Notons que cette hypothèse sur l'algorithme 2.1 est très simple et intuitivement logique, la région de confiance devant s'élargir si l'itération est très réussie, à moins qu'elle ne soit déjà suffisamment grande. La mise à jour générale

$$\Delta^{(k+1)} \in \begin{cases} [\,\gamma_1 \Delta^{(k)}, \gamma_2 \Delta^{(k)}\,] & \text{si } \rho^{(k)} < \eta_1, \\ [\,\gamma_2 \Delta^{(k)}, \Delta^{(k)}\,] & \text{si } \rho^{(k)} \in [\eta_1, \eta_2[, \\ [\,\gamma_3 \Delta^{(k)}, \gamma_4 \Delta^{(k)}\,] & \text{si } \rho^{(k)} \geq \eta_2. \end{cases} \tag{4.38}$$

avec

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{et} \quad 0 < \gamma_1 \leq \gamma_2 < 1 < \gamma_3 \leq \gamma_4, \tag{4.39}$$

satisfait à l'hypothèse 4.13 sans faire intervenir de $\Delta_{\max}$[15].

[15]D'un point de vue numérique toutefois, $\Delta_{\max}$ existera bel et bien ; il est sage de ne pas laisser le rayon de confiance s'accroître de façon immodérée.

4.3.4 Théorème de convergence globale.

Les dernières hypothèses formulées ci-avant permettent d'établir le théorème de convergence suivant.

Théorème 4.6 *Si les hypothèses 4.1–4.13 sont satisfaites et si x^* est un point limite de la suite d'itérés $\{x^{(k)}\}$, alors x^* est un point critique du second ordre.*

Le résultat de ce théorème est plus fort que celui du théorème 4.5 au prix d'hypothèses supplémentaires relativement peu contraignantes. L'avantage d'utiliser des approximations locales non forcément convexes est donc évident. La démonstration de ce résultat remarquable peut être trouvée dans [20]. Notons qu'un point critique du second ordre ne signifie pas forcément minimum local : les points d'inflexion multidimensionnels vers lesquels l'algorithme peut converger ne peuvent être évités qu'en tenant compte d'informations d'ordre plus élevé que le second. Néanmoins, si la matrice hessienne obtenue au point limite est strictement définie positive[16], on est assuré que ce point est un minimum local.

4.4 Forme des régions de confiance.

La norme $\|\cdot\|_k$ définit la forme de la région de confiance

$$\mathcal{B}^{(k)} = \left\{ x \in \mathbb{R}^n : \ \|x - x^{(k)}\|_k \leq \Delta^{(k)} \right\} \tag{4.40}$$

à l'itération k. La norme euclidienne classique (ou norme ℓ_2) lui donne la forme d'une sphère alors que les normes ℓ_1 et ℓ_∞ lui donnent la forme d'un cube. La figure 4.3 illustre la forme des régions de confiance dans un espace à deux dimensions pour un même rayon de confiance $\Delta^{(k)}$ avec les normes de type ℓ_p

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}. \tag{4.41}$$

En prenant la limite pour p tendant vers l'infini, on a

$$\|x\|_\infty = \max_{i \in \{1,2,\ldots,n\}} |x_i|. \tag{4.42}$$

L'utilisation de la norme ℓ_∞ correspond à de simples contraintes de bornes variant d'une itération à l'autre, celles-ci sont parfois appelées « move limits » dans le domaine de l'optimisation des structures (voir par exemple [9]).

[16] D'un point de vue numérique, la définie-positivité doit naturellement être évaluée à une certaine tolérance près.

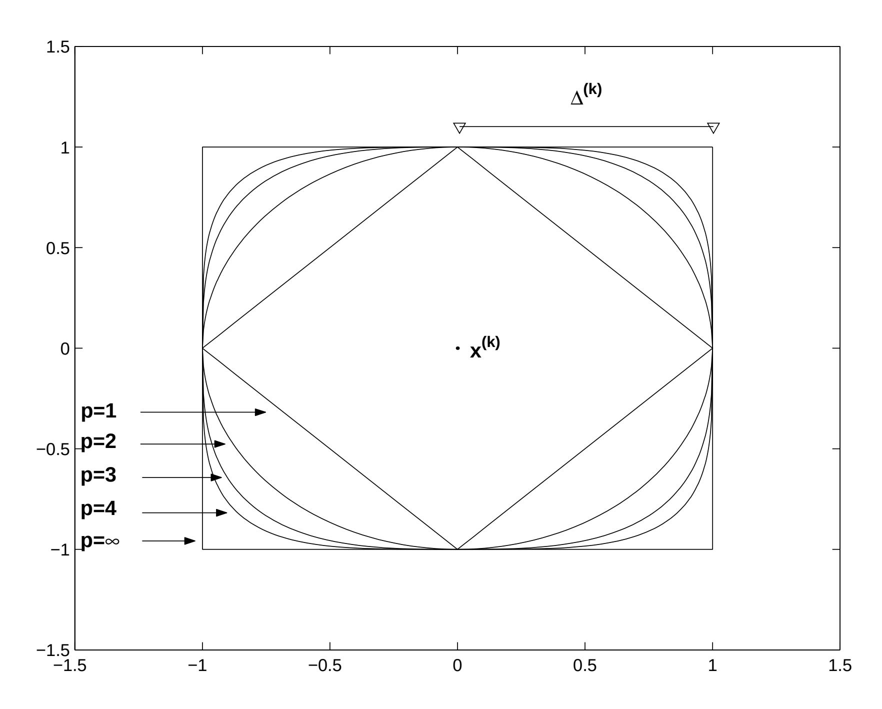


FIG. 4.3 – Forme des régions de confiance dans un espace à deux dimensions avec les normes ℓ_p pour un même rayon de confiance $\Delta^{(k)}$.

Les régions de confiance prennent une forme ellipsoïdale pour une norme matricielle

$$\|x\|_M = \sqrt{x^T M x} \tag{4.43}$$

où $M \in \mathbb{R}_n^n$ est une matrice symétrique définie positive. Les axes principaux de l'ellipsoïde sont les directions correspondant aux vecteurs propres de la matrice M alors que l'étendue de l'ellipsoïde dans cette direction est proportionnelle à l'inverse de la valeur propre correspondante (voir figure 4.4).

Il est assez fréquent que, dans un problème pratique, les variables aient des ordres de grandeurs sensiblement différents. Si l'ordre de grandeur d'une des variables, disons x_1 pour fixer les idées, est nettement supérieur à celui des autres variables, le problème risque d'être mal conditionné. En effet, lors du calcul du pas de progression $s^{(k)}$, la contrainte de confinement au sein de la région de confiance s'exprimerait comme

$$\|s^{(k)}\| \simeq |s_1^{(k)}| \leq \Delta^{(k)} \tag{4.44}$$

faisant ainsi perdre toute information sur les autres variables. De plus, l'algorithme se retrouve dans l'impossibilité de limiter raisonnablement les déplacements sur ces mêmes variables, rendant de ce fait les propriétés de convergence caduques.

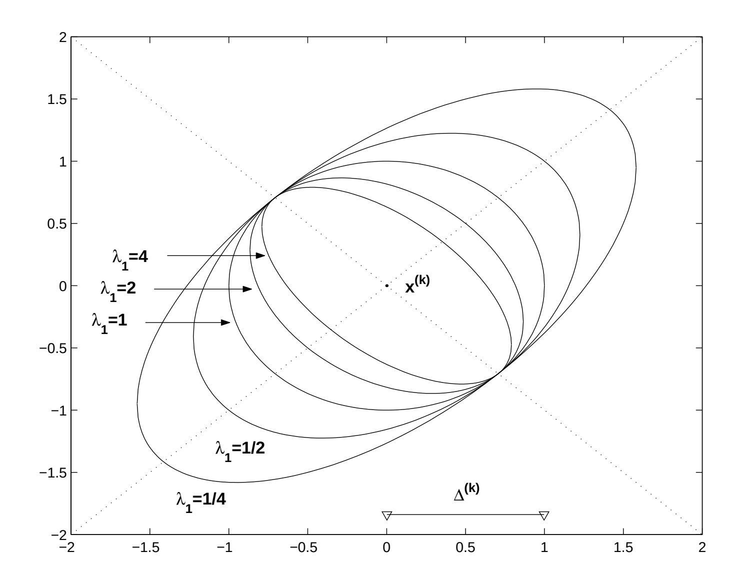


FIG. 4.4 – Forme des régions de confiance dans un espace à deux dimensions avec une norme matricielle pour un même rayon de confiance $\Delta^{(k)}$. Les directions en pointillé correspondent aux vecteurs propres de la matrice symétrique définie positive M. Les régions de confiance sont dessinées pour différentes valeurs de la valeur propre λ_1 tandis que la seconde valeur propre est maintenue constante et égale à l'unité.

Il est donc d'une importance capitale de normaliser le problème de manière adéquate.

En plus du danger numérique de travailler avec des variables de trop grande valeur absolue, celles-ci peuvent avoir des échelles de variation complètement différentes.

Pour lever ces difficultés, nous pouvons écrire chacune des variables de la manière suivante :

$$x_i = x_i^{\text{car}} + \xi_i \, \delta x_i^{\text{car}} \tag{4.45}$$

où x_i^{car} est une valeur caractéristique, δx_i^{car} une échelle de variation caractéristique et ξ_i la variable normalisée. Dès lors, le problème d'optimisation peut être résolu en terme des ξ_i, ce qui permet de travailler à des ordres de grandeur raisonnables et avec des plages de variation pour les variables de contrôle d'amplitudes comparables.

En terme de régions de confiance, ceci se traduit fort simplement. Une région de confiance sphérique dans l'espace des ξ_i correspond, dans l'espace des x_i, à l'utilisation d'une norme matricielle (4.43) où

$$M = \text{diag}\left(\frac{1}{\delta x_1^{\text{car}}}, \ \ldots, \ \frac{1}{\delta x_n^{\text{car}}}\right) \tag{4.46}$$

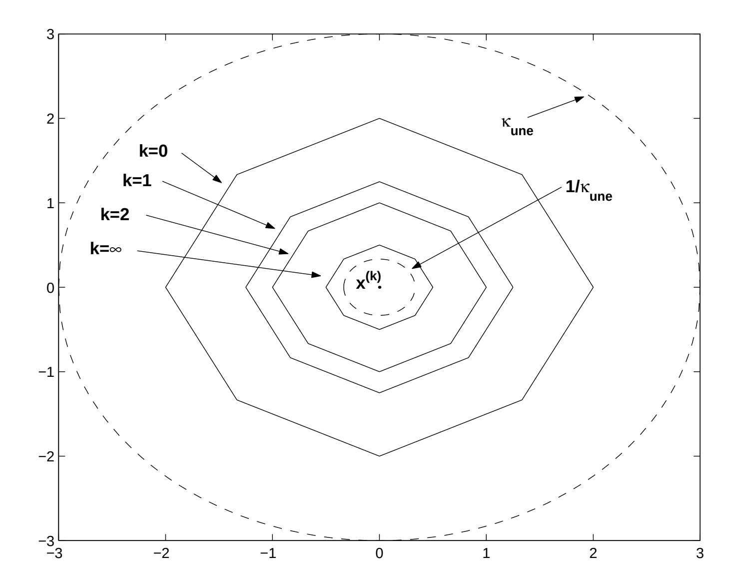


FIG. 4.5 – Normes uniformément équivalentes à la norme euclidienne : illustration avec la norme (4.47) pour différentes valeurs de k et pour $\Delta^{(k)} = 1$.

i.e. une région de confiance ellipsoïdale d'autant plus étirée dans une direction que la variation caractéristique de la variable correspondante est faible.

La possibilité de changer de norme d'une itération à l'autre pour définir la région de confiance $\mathcal{B}_k$ doit cependant être utilisée prudemment. C'est l'hypothèse 4.9 qui assure cette « prudence » : elle définit un rayon maximal et un rayon minimal (en norme euclidienne) pour la frontière de la région de confiance, ceux-ci sont proportionnels au rayon de confiance à l'itération courante $\Delta^{(k)}$. La figure 4.5 illustre le concept pour la norme

$$\|x\|_k = \frac{k+1}{k+4}\left(\sum_{i=1}^{n}|x_i| + \max_{i=1}^{n}|x_i|\right). \tag{4.47}$$

qui est uniformément équivalente à la norme euclidienne avec $\kappa_{une} = 3$.

4.5 Problèmes non-différentiables.

Dans cette section, nous envisageons le cas où la fonction objectif $f(x)$ est continue mais pas nécessairement différentiable en tout point de son domaine de définition. Le tableau est brossé rapidement et ne présente que le strict nécessaire, le lecteur intéressé est invité à se reporter à l'ouvrage de Conn, Gould et Toint [20] dans lequel il trouvera de plus amples détails.

Pour un problème non-différentiable et non-contraint, nous avons la condition nécessaire d'optimalité suivante pour x^*

$$0 \in \partial f(x^*) \tag{4.48}$$

pour autant que $f(x)$ soit localement continue au sens de Lipschitz[17]. Un point qui satisfait (4.48) est un *point critique du premier ordre* de $f(x)$. Pour rappel, quelques notations concernant les problèmes non-différentiables ont été introduites à la section 3.1.

Un cas particulier important est celui des fonctions non-différentiables *composées* de la forme

$$f(x) = \phi_0(x) + h(\phi_1(x), \ldots, \phi_m(x)) \tag{4.50}$$

où $\phi_i : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ pour tout $i = 0, \ldots, m$ sont des fonctions convexes continûment dérivables et $h : \mathcal{C} \subseteq \mathbb{R}^m \to \mathbb{R}$ une fonction convexe et continue au sens de Lipschitz. Dans ce cas, le sous-différentiel se calcule aisément

$$\partial f(x) = \left\{ \nabla_x \phi_0(x) + \sum_{i=1}^{m} y_i \nabla_x \phi_i(x) \ : \ y \in \partial h(\phi_1(x), \ldots, \phi_m(x)) \right\} \tag{4.51}$$

et la condition nécessaire du premier ordre (4.48) s'écrit

$$\nabla_x \phi_0(x^*) + \sum_{i=1}^{m} y_i^* \nabla_x \phi_i(x^*) = 0. \tag{4.52}$$

pour $x^* \in \mathcal{D}$ et $y^* \in \partial h(\phi_1(x^*), \ldots, \phi_m(x^*))$.

De manière assez surprenante (voir [20]), le schéma décrit par l'algorithme 2.1 reste adéquat pour traiter des problèmes non-différentiables moyennant l'utilisation de la formule de mise à jour du rayon de confiance (4.38) assortie de la même condition (4.39) à laquelle nous ajoutons la contrainte

$$\frac{1}{\gamma_3} < \gamma_1. \tag{4.53}$$

Certaines hypothèses minimales doivent cependant être faites sur le problème. La fonction objectif doit être localement continue au sens de Lipschitz et *régu-*

---

[17]Une fonction $f(x) : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ est continue au sens de Lipschitz s'il existe une constante κ telle que, pour tout $x, y \in \mathcal{D}$,

$$|f(x) - f(y)| \leq \kappa |x - y|. \tag{4.49}$$

La fonction $f(x)$ est *localement* continue au sens de Lipschitz si, pour tout $x \in \mathcal{D}$, il existe un voisinage de x dans lequel $f(x)$ est continue au sens de Lipschitz.

lière[18] sur $\mathbb{R}^n$. Formellement, cette première hypothèse sur la forme du problème s'écrit comme suit.

Hypothèse 4.14 *La fonction $f(x)$ est localement continue au sens de Lipschitz et régulière sur $\mathbb{R}^n$.*

Malgré l'intérêt certain que présente cette classe de fonctions, nous nous limiterons la plupart du temps aux seules fonctions *convexes* satisfaisant à l'hypothèse 4.14.

Dans l'expression de l'algorithme 2.1 il faut naturellement redéfinir l'approximation locale et ce que nous entendons par une réduction « suffisante » de l'approximation locale. Notre approximation locale doit cependant être un peu moins générale que dans le cas différentiable. En effet, nous adoptons une approximation locale $m(x,p,s)$ qui dépend d'un ensemble de paramètres $p \in \mathcal{P}$ pouvant être ajustés à chaque itération. Entre autres exemples, les paramètres p peuvent être des multiplicateurs de Lagrange, un facteur de pénalité ou des approximations des dérivées d'ordre supérieur (lorsque celles-ci existent).

Hypothèse 4.15 *L'approximation locale $m(x,p,s)$ est localement continue au sens de Lipschitz et régulière par rapport à s pour tout $(x,p) \in \mathbb{R}^n \times \mathcal{P}$ et continue en (x,p) pour tout $s \in \mathbb{R}^n$.*

L'approximation locale $m(x,p,s)$ doit donc varier continûment par rapport aux paramètres p qui sont les seuls à être modifiés en cas d'itération infructueuse. Ceci est une restriction par rapport au cas différentiable où l'approximation $m^{(k)}(x^{(k)}+s)$ pouvait changer indépendamment en passant d'une itération à l'autre. Cette première hypothèse sur $m(x,p,s)$ est le pendant non-différentiable de l'hypothèse 4.4. De façon assez logique, nous devons également formuler les hypothèses suivantes, qui sont semblables à 4.5 et 4.6. Elles garantissent que l'approximation locale est au moins une approximation du premier ordre de la fonction objectif.

Hypothèse 4.16 *L'approximation locale et la fonction objectif coïncident quand $s = 0$, i.e.*

$$m(x,p,0) = f(x) \tag{4.55}$$

pour tout $(x,p) \in \mathbb{R}^n \times \mathcal{P}$.

---

[18] Une fonction est régulière si sa dérivée directionnelle (3.4) $f'_d(x)$ existe pour tout x et $d \in \mathbb{R}^n$ et si celle-ci correspond à la *dérivée directionnelle généralisée* de f en x et dans la direction d qui se définit par la limite

$$\lim_{t \to 0^+} \sup_{y \to x} \frac{f(y+td) - f(y)}{t}. \tag{4.54}$$

Hypothèse 4.17 *Les dérivées directionnelles de l'approximation locale et de la fonction objectif coïncident pour toute direction non-nulle d quand $s = 0$, i.e.*

$$m'_d(x, p, 0) = f'_d(x) \tag{4.56}$$

pour tout $d \neq 0 \in \mathbb{R}^n$ et pour tout $(x, p) \in \mathbb{R}^n \times \mathcal{P}$.

L'hypothèse suivante est essentiellement technique. Elle permet de dégager des propriétés de convergence intéressantes.

Hypothèse 4.18 *L'ensemble des paramètres $\mathcal{P}$ est fermé et borné.*

De manière analogue au raisonnement utilisé pour introduire l'hypothèse 4.8, il nous faut spécifier des conditions auxquelles doit répondre le pas $s^{(k)}$ pour effectuer une « réduction *suffisante* de l'approximation locale ». Un point de Cauchy peut être défini de façon similaire au cas différentiable. Il amène à poser l'hypothèse suivante sur le pas de progression $s^{(k)}$.

Hypothèse 4.19 *Le pas de progression $s^{(k)}$ est tel que, pour tout x^* donné,*

$$m(x^{(k)}, p^{(k)}, 0) - m(x^{(k)}, p^{(k)}, s^{(k)}) \geq \kappa_{mdc} \|g^{(k)}\| \min(\delta, \Delta^{(k)}) \tag{4.57}$$

lorsque $\|x^{(k)} - x^*\| < \varepsilon$. Le vecteur $g^{(k)}$ est le sous-gradient (3.8) *de norme minimum et la constante $\kappa_{mdc} \in]0, 1[$. Les deux constantes strictement positives δ et ε sont choisies en fonction de x^*.*

Moyennant toutes ces hypothèses et modifications, nous pouvons obtenir le théorème de convergence suivant pour l'algorithme 2.1.

Théorème 4.7 *Si x^* est un point limite de la suite $\{x^{(k)}\}$ et si les hypothèses 4.14 à 4.19 sont satisfaites, alors x^* est un point critique du premier ordre de $f(x)$.*

Cet énoncé n'est valable que pour une région de confiance définie avec la norme euclidienne ; pour d'autres normes, il convient d'ajouter l'hypothèse 4.9. Remarquons que, contrairement au théorème 4.4, son pendant différentiable, le théorème 4.7, *suppose* qu'il existe un point limite à la suite $\{x^{(k)}\}$.

4.6 Conclusion.

Ce chapitre traite de propriétés très générales des algorithmes utilisant une globalisation par régions de confiance. Les hypothèses sont détaillées et présentées avec rigueur afin de donner une solide assise théorique aux algorithmes développés dans la suite de ce travail. Plusieurs sujets sont évoqués. Les hypothèses

permettant de conclure à la convergence vers un point critique du premier ordre ou du second ordre sont clairement exposées pour une fonction objectif différentiable ainsi que les précautions importantes concernant la forme des régions de confiance. Tout ce *corpus* servira de base à l'implémentation concrète détaillée dans le chapitre 6.

Le présent chapitre traite également des fonctions non-différentiables. En effet, l'implémentation d'un algorithme pour des problèmes d'optimisation avec contraintes nous a amené à utiliser des fonctions de mérite non-différentiables et les propriétés correspondantes se sont donc avérées extrêmement utiles.

Le propos et le ton de ce chapitre sont volontairement très généraux, nous permettant ainsi de laisser une certaine place à la créativité dans l'implémentation concrète d'un algorithme tout en préservant des propriétés de convergence claires. Le cadre est riche et son exploitation complète pour la construction d'algorithmes performants constitue un travail qui est loin d'être achevé.

Deuxième partie

Optimisation non-contrainte

Chapitre 5

L’identification des paramètres d’un modèle dynamique

Parmi les problèmes d’optimisation sans contrainte, certains occupent une place privilégiée de par leur forme particulière et leur fréquence (consulter, par exemple, l’ouvrage de Schittkowski [90]). Il s’agit des problèmes d’*identification* ou de *calibration* des paramètres d’un modèle. Le présent chapitre évoque la question, justifie son importance, pose le cadre théorique et envisage une application pratique sur un modèle dynamique.

5.1 Modélisation.

Dans le langage courant, un *modèle* désigne aussi bien un objet sur lequel il convient de conformer son comportement qu’un abrégé de toutes les qualités. Le modèle est donc tout à la fois moule, gabarit, prescription, résumé et réduction. Il porte soit sur un processus qu’il aide à retranscrire, soit sur un objet dont il contracte les propriétés.

Utilisé abondamment dans les sciences et les techniques, le modèle demeure un intermédiaire indispensable : il s’interpose entre les phénomènes et l’interprétation que la science en donne. En particulier, dans les sciences de la nature, le modèle apparaît comme nécessaire pour affronter le réel qui se révèle bien souvent peu accessible à l’expérience immédiate et, dans la plupart des cas, trop complexe pour être appréhendé [91].

Les ingrédients initiaux d’un modèle sont, d’une part, tout ce qui a pu être observé et mesuré sur un objet particulier, d’autre part, toutes les connaissances théoriques de la physique, de la biologie, de la sociologie, etc., qu’il est possible de mobiliser. Construit par un agencement judicieux de ces ingrédients, le modèle donne une image simplifiée et malléable à loisir qui est émise à titre d’hypothèse

pour la confronter aux données issues d'observations ou d'expériences afin de la valider. En somme, le modèle présente à l'intuition une explication intelligible et partielle d'un phénomène trop complexe pour être saisi sous tous ses aspects. Sa fonction est double : il résume des connaissances empiriques que l'on possède sur un objet particulier et il permet d'éprouver ces connaissances en vue de prévoir un comportement.

5.1.1 Modélisation mathématique.

Dans le cadre des applications des mathématiques, on utilise couramment le *modèle mathématique* pour désigner une équation ou un système d'équations destiné à représenter le phénomène étudié ; on parle alors de « modélisation » pour signifier « mise en équations » [91]. On décrit le système réel au moyen de variables d'état dont l'évolution est gouvernée par des équations d'évolution. Le modèle se substitue en quelque sorte à la réalité si bien que les conclusions sont le résultat de l'analyse de celui-ci plutôt que du système réel.

D'après Nihoul [79], un modèle idéal serait nécessairement quadridimensionnel (trois variables spatiales et le temps) et comporterait une infinité de variables d'état. Ce modèle cependant ne serait rien d'autre que la nature elle-même, à l'image de ce plan si précis qu'il recouvre entièrement le lieu qu'il détaille. L'envergure d'un modèle mathématique est limitée, notamment, par les nécessités de sa *calibration* : il faut disposer de suffisamment de données pour imposer de bonnes conditions initiales et aux limites et pour déterminer les paramètres numériques intervenant dans sa formulation. C'est ce dernier point qui constitue précisément l'objet de ce chapitre.

5.1.2 Identification paramétrique.

Toute modélisation mathématique d'une réalité physique passe par l'introduction d'une série de paramètres. Ces paramètres sont plus ou moins bien connus suivant les cas : la viscosité d'un fluide peut être mesurée expérimentalement ou déterminée de manière théorique, il en va de même pour le module de Young d'un matériau élastique, etc. Les modèles sont toutefois confrontés à des contraintes (coût du calcul numérique, nécessité d'interpréter les résultats, de les représenter de façon appropriée, ...) qui conduisent à limiter l'envergure soit par la réduction de l'ampleur (en ne considérant, par exemple, que des valeurs moyennes sur la profondeur, sur la section droite d'une rivière, une zone climatique, une région économique ...), soit par sectorisation (en se limitant à un sous-modèle écologique, économique ...), soit par agrégation (en se limitant aux caractéristiques globales d'ensembles de variables d'état : la concentration moyenne des particules

en suspension dans l'air ou dans l'eau, sans distinction de tailles ou de compositions ; les céréales, sans distinction d'espèces ; le nombre de chômeurs, sans distinction de catégories ...). Ces limitations de l'envergure du modèle conduisent inévitablement à une modélisation empirique des phénomènes à simplifier ou à agréger, les paramètres introduits à ce niveau sont peu connus et difficilement mesurables : il s'agit alors de choisir les valeurs qui seront les plus adéquates. Pour ce faire, la méthode classique implique la répétition d'un très grand nombre de simulations et la comparaison des résultats obtenus avec les observations. Lorsque le nombre de paramètres est important, cette procédure est erratique et son issue dépend fortement de l'expérience et du talent du modélisateur. L'objet de ce travail est de systématiser cette recherche d'un ensemble optimal de paramètres pour un modèle donné.

Les techniques adoptées sont diverses. La technique utilisée ici est celle de l'optimisation : il s'agit de définir une fonction objectif qui représente l'écart entre les résultats du modèle et les mesures et de minimiser celle-ci dans l'espace des paramètres.

5.1.3 Traitement des résultats du modèle.

Il arrive souvent que les valeurs à comparer ne soient pas les variables du modèle elles-mêmes mais une information tirée d'une combinaison de celles-ci qui sera confrontée à cette même information déduite des observations : si nous voulons un modèle qui donne une bonne approximation de la vitesse moyenne d'un fluide dans une canalisation, il n'est pas nécessaire de comparer les vitesses *mesurées* en chaque point de la conduite et les vitesses *calculées* en chaque point de la conduite, d'autant plus qu'en général les points de mesure et les noeuds du maillage ne sont pas rigoureusement identiques. Un certain post-traitement des résultats et des mesures est donc généralement nécessaire. L'interprétation des résultats se fait par le biais de fonctions ou de valeurs discrètes que nous nommerons les *valeurs de comparaison*

$$c_j \qquad j = 1, \ldots, m. \tag{5.1}$$

Celles-ci sont donc obtenues après traitement des variables d'état

$$s_i(t) \qquad i = 1, \ldots, N_s \tag{5.2}$$

où t est le vecteur des variables indépendantes (généralement le temps et l'espace). Les variables d'état proviennent de la résolution (numérique ou analytique) des équations du modèle[1]. Formellement,

$$c_j = R_j(s_1, \ldots, s_{N_s}, t) \qquad j = 1, \ldots, m. \tag{5.3}$$

[1] En cas de résolution numérique, les variables d'état se présentent sous la forme d'un ensemble de valeurs discrètes dont le nombre est très nettement supérieur au nombre de variables d'état lui-

Notons que chaque s_i et chaque c_j peut être une fonction ou une variable discrète. Nous nommerons les fonctions R_j les *fonctions de traitement des résultats*.

5.1.4 Traitement des mesures.

Il n'est pas toujours aisé de mesurer ce qui nous serait le plus utile, certaines mesures sont parfois complètement inaccessibles. Beaucoup se font ainsi de manière indirecte : on mesure certaines quantités dont on déduit les valeurs intéressantes.

Soit un ensemble de mesures

$$m_k \qquad k = 1, \ldots, N_m. \tag{5.4}$$

Celles-ci peuvent être des fonctions ou des valeurs discrètes (ce qui est nettement plus fréquent). Pour pouvoir comparer le modèle et la réalité, il convient de générer des *valeurs de référence*

$$\hat{c}_j \qquad j = 1, \ldots, m \tag{5.5}$$

auxquelles nous comparerons les valeurs c_j issues du modèle mathématique. Formellement,

$$\hat{c}_j = M_j(m_1, \ldots, m_{N_m}) \qquad j = 1, \ldots, m \tag{5.6}$$

où les M_j sont des fonctions ou des fonctionnelles suivant la nature des m_k. Notons que, tout comme les valeurs de comparaison, les $\hat{c}_j$ peuvent être des fonctions et/ou des variables discrètes. Nous nommerons les fonctions M_j les *fonctions de traitement des mesures*.

5.2 Caractère mal posé du problème.

De manière générale, le problème d'optimisation paramétrique est un problème inverse et, par ce fait, mal posé [57, 77]. En effet, un problème est dit *bien posé* lorsqu'il respecte les conditions d'existence, d'unicité et de continuité de la solution pour tout jeu de données. Cependant, dans les problèmes de calibration, la modélisation forcément réductrice, les erreurs numériques, les mesures bruitées, etc. mènent inévitablement à l'inexistence d'une solution conduisant parfaitement au résultat souhaité.

---

même. Par souci de simplicité, nous conserverons néanmoins la notation s_i pour cet ensemble de valeurs discrètes (N_s sera dès lors considérablement plus grand).

5.2.1 Quasi-solution d'un problème inverse.

On introduit la méthode de régularisation suivante. Dans le cas où les paramètres sont incompatibles, c'est-à-dire qu'aucun ensemble de valeurs de ceux-ci ne peut conduire exactement à la réponse désirée, on cherche l'ensemble de paramètres le plus prochede celle-ci au sens d'une certaine distance. Ce jeu de paramètres est appelé *quasi-solution* du problème inverse. En changeant de cette façon le concept de solution, on est assuré de son existence.

Les problèmes inverses et, en particulier, celui de l'optimisation paramétrique, peuvent donc se mettre sous la forme canonique d'un problème d'optimisation dans lequel une distance est définie et doit être minimisée en jouant sur les grandeurs choisies comme variables de contrôle. Mathématiquement, il s'agit de minimiser une fonction objectif $\mathcal{F}(c,\hat{c})$, qui représente la distance entre les résultats du modèle c et les mesures correspondantes $\hat{c}$ (après un éventuel traitement), en faisant varier c par le biais d'une modification des paramètres

$$x_\ell \qquad \ell = 1,\ldots,n \tag{5.7}$$

du modèle. Les $\hat{c}_j$ étant des *valeurs de référence* fixes, nous pouvons réécrire la fonction objectif sous la forme $f(x) = \mathcal{F}(c(x),\hat{c})$ où x est le vecteur des paramètres du modèle ou variables de contrôle. Le problème se formule finalement comme suit :

$$\text{trouver } x^* = \arg\min_x f(x) \tag{5.8}$$

qui est la forme canonique d'un problème d'optimisation non-contraint.

5.2.2 Quantification de l'erreur : fonction objectif.

Soient deux ensembles de grandeurs c_j et $\hat{c}_j$, $j = 1,\ldots,m$, celles-ci pouvant être des fonctions ou des valeurs discrètes, c est le *vecteur des valeurs de comparaison* et $\hat{c}$ le *vecteur des valeurs de référence* résultant respectivement du traitement des résultats et du traitement des mesures. Nous désirons une fonction objectif qui mesure l'écart entre ces deux vecteurs ; $\mathcal{F}(c,\hat{c})$ doit donc avoir les propriétés d'une distance [68]

1. Positivité : $\mathcal{F}(x,y) \geq 0$ pour tout couple x, y de $\mathbb{R}^m$.
2. Séparation : $\mathcal{F}(x,y) = 0$ si et seulement si $x = y$.
3. Symétrie : $\mathcal{F}(x,y) = \mathcal{F}(y,x)$.
4. Inégalité du triangle : $\mathcal{F}(x,y) \leq \mathcal{F}(x,z) + \mathcal{F}(z,y)$ pour tout x, y et z de $\mathbb{R}^m$.

Parmi les fonctions répondant à ces critères, on peut citer quelques l'écart au sens des moindres carrés ou les normes pondérées de différents.

5.2.2.1 Écart au sens des moindres carrés.

L'exemple le plus simple (c_j tous discrets) est celui de la *norme euclidienne* ou erreur *au sens des moindres carrés* :

$$\mathcal{F}(c,\hat{c}) = \frac{1}{2}\|c-\hat{c}\|^2 = \frac{1}{2}(c-\hat{c})^T(c-\hat{c}) = \sum_{i=1}^{m}(c_i-\hat{c}_i)^2. \tag{5.9}$$

L'introduction d'une matrice diagonale D permet de tenir compte d'éventuelles différences d'ordre de grandeur ou d'importance

$$\mathcal{F}(c,\hat{c}) = \frac{1}{2}\|c-\hat{c}\|_D^2 = \frac{1}{2}(c-\hat{c})^T D(c-\hat{c}) = \sum_{i=1}^{m} D_{ii}(c_i-\hat{c}_i)^2. \tag{5.10}$$

Mais la forme la plus générale prend en considération des corrélations croisées :

$$\mathcal{F}(c,\hat{c}) = \frac{1}{2}\|c-\hat{c}\|_W^2 = \frac{1}{2}(c-\hat{c})^T W(c-\hat{c}) \tag{5.11}$$

où W est idéalement l'inverse de la matrice de covariance des erreurs sur les observations [60, 77, 94].

5.2.2.2 Norme pondérée d'ordre p.

L'écart entre deux vecteurs peut également être mesurée au moyen des normes pondérées d'ordre p

$$\mathcal{F}(c,\hat{c}) = \left\{\sum_{i=1}^{m} w_i\,|c_i-\hat{c}_i|^p\right\}^{1/p} \tag{5.12}$$

où w_i est un ensemble de coefficients de pondération positifs.

De la même manière, si $c(t)$ et $\hat{c}(t)$ sont des fonctions de la variable indépendante t, $\mathcal{F}$ est une fonctionnelle

$$\mathcal{F}(c,\hat{c}) = \left\{\int_0^T w(t)\,|c(t)-\hat{c}(t)|^p\,dt\right\}^{1/p} \tag{5.13}$$

où $w(t)$ est une fonction de pondération positive.

5.2.3 Analyse de l'optimum.

En plus de fournir une quasi-solution au problème inverse de l'identification paramétrique, le processus d'optimisation peut également fournir d'autres informations tout aussi intéressantes. Par exemple, certaines méthodes d'optimisation

travaillent avec des approximations successives de la matrice hessienne de plus en plus précises. A la fin du calcul, l'inspection de celle-ci fournit une analyse d'erreur et de sensibilité des paramètres, ce qui nous dispense du fastidieux travail qui consiste à perturber chacun des paramètres indépendamment et d'en analyser les effets sur les résultats [71].

En effet, lorsque les erreurs dans les observations sont supposées être normalement distribuées, des intervalles d'incertitude peuvent être obtenus en analysant la matrice hessienne. Le développement en série de Taylor de la fonction objectif autour de la solution optimale x^* donne

$$f(x) \simeq f(x^*) + \frac{1}{2}(x - x^*)^T H (x - x^*), \tag{5.14}$$

où $H = \nabla_{xx} f(x^*)$ est la matrice hessienne à l'optimum. Si les termes négligés sont suffisamment petits, les incertitudes sur les paramètres optimaux du modèle sont normalement distribués avec une moyenne nulle et une matrice de covariance définie comme l'inverse du Hessien, $C = H^{-1}$. Cette matrice fournit donc une information sur la distribution de probabilité des paramètres optimaux. Les éléments de la diagonale de C fournissent une mesure de la largeur de la distribution pour chacun des paramètres. Ceci remplace avantageusement une analyse de sensibilité classique. De plus, la matrice de covariance fournit des informations supplémentaires, les termes hors-diagonale indiquant le niveau de corrélation entre deux paramètres du modèle.

5.3 Analyse de l'observabilité par expérience jumelle.

Une expérience jumelle est une simulation par le modèle d'une prise de mesures. Pour chaque paramètre du modèle mathématique, une valeur de référence est choisie qui permettra d'effectuer une simulation du modèle mathématique. Des valeurs ainsi obtenues pour les variables d'état, on extrait des données semblables aux mesures dont on dispose pour le système réel.

Cette expérience nous permet de tirer des conclusions sur l'observabilité du système. Est-ce que les mesures à notre disposition sont suffisantes pour en déduire les paramètres optimaux du modèle ? Elle nous donne aussi accès à une analyse de l'influence du bruit sur l'assimilation des données.

La réalisation d'une expérience jumelle consiste à trouver une fonction qui générera des valeurs de mesures m_k fictives à partir des variables d'état du modèle (qui auront été générées avec un ensemble de paramètres de référence x^{ref}) à laquelle on ajoute une fonction simulant des erreurs de mesures

$$m_k^{\mathrm{ref}} = T_k(s_1, \ldots, s_{N_s}, t) + B_k \qquad k = 1, \ldots, N_m. \tag{5.15}$$

Nous appellerons T_k les *fonctions jumelles* et B_k les *fonctions de bruit*.

Nous devons ensuite utiliser ces mesures avec la procédure d'assimilation de données. La procédure idéale est évidemment celle qui nous rend pour les paramètres optimaux ceux qui ont servi à la génération des mesures fictives, à savoir x^{ref}.

Théorème 5.1 *En supposant qu'il n'y a pas d'erreur de mesure ($B_k = 0$), l'ensemble de paramètres optimaux x^* sera l'ensemble de paramètres de référence x^{ref} si et seulement si la fonction objectif $\mathcal{F}$ présente un minimum global au point $x = x^{ref}$. Au vu de la définition d'une distance $\mathcal{F}$, la valeur de $\mathcal{F}$ à ce minimum est* 0 *et elle est atteinte si et seulement si*

$$c = \hat{c}. \tag{5.16}$$

Ceci entraîne, au vu des définitions (5.3) et (5.6) que

$$M_j\left(T_1(s_1,\ldots,s_{N_s},t),\ldots,T_{N_m}(s_1,\ldots,s_{N_s},t)\right) = R_j\left(s_1,\ldots,s_{N_s},t\right) \tag{5.17}$$

pour $j = 1,\ldots,m$.

Cette relation lie les fonctions T, R et M. Notons que les fonctions T nous sont imposées par les mesures dont nous disposons (ou dont nous voudrions analyser l'observabilité), ce qui signifie que le choix de R conditionne complètement celui de M et inversement. Il est intéressant de noter que la notion théorique d'expérience jumelle permet à elle seule de trouver une relation entre les fonctions de traitement des résultats et de traitement des mesures qui doit être vérifiée même si aucune expérience jumelle n'est réellement pratiquée.

Une expérience jumelle peut servir de base pour analyser l'observabilité du système. Elle permet de répondre à la question suivante : « Quelles mesures dois-je effectuer sur le système afin d'en déterminer les paramètres numériques ? ». En effet, il suffit pour cela d'effectuer l'expérience jumelle avec un ensemble de « mesures » de notre choix et de constater quels sont les paramètres qui sont recouvrés. On peut se servir de cette information avant une campagne de mesure afin de distinguer celles qui sont nécessaires à une bonne calibration du modèle choisi et celles qui sont redondantes voire inutiles.

Le concept d'expérience jumelle permet également de tester la robustesse de la méthode de calibration par rapport à des erreurs de mesure. Il suffit de bruiter les mesures simulées et d'interpréter le recouvrement des paramètres : les différents paramètres convergeront soit vers une valeur proche de celle qui a servi à l'expérience jumelle, la méthode est dans ce cas dite *robuste* pour ce paramètre, soit vers une autre valeur et la méthode est alors dite *non robuste*.

5.4 Différentiation de la fonction objectif.

Comme nous l'avons vu dans les chapitres précédents, certaines méthodes d'optimisation (celles requérant le moins d'évaluations de la fonction objectif) exigent l'évaluation du gradient

$$\nabla_x f(x^{(k)}) \tag{5.18}$$

de cette fonction par rapport aux variables de contrôle ou de la matrice jacobienne

$$J(x^{(k)}) = \frac{\partial c(x^{(k)})}{\partial x}. \tag{5.19}$$

La qualité de la convergence de ces méthodes dépend de la précision du calcul de ces dérivées. Suivant les applications, différentes méthodes de différentiation sont disponibles (voir à ce sujet [57, 101]). La plus simple à mettre en œuvre est la méthode des différences finies. D'autres procédés de différentiation automatique existent néanmoins : la différentiation directe et le modèle adjoint [86].

5.4.1 Méthode des différences finies.

C'est la méthode la plus simple à mettre en œuvre puisqu'il suffit de perturber chaque paramètre x_i successivement positivement puis négativement d'une valeur à définir δ_i, de calculer dans chaque cas la valeur de la fonction objectif et d'approcher les dérivées souhaitées par le quotient différentiel, soit pour le gradient de la fonction objectif

$$\frac{\partial f(x^{(k)})}{\partial x_i} = \frac{f(x^{(k)} + \delta_i e_i) - f(x^{(k)} - \delta_i e_i)}{2\,\delta_i} + o(\delta_i^2) \tag{5.20}$$

où les e_i sont les vecteurs de la base canonique de $\mathbb{R}^{N_x}$. Cette méthode centrée nécessite $2N_x$ évaluations de la fonction objectif pour le calcul du gradient en plus de l'évaluation de la valeur de la fonction, augmentant d'autant le temps de calcul d'une itération.

De la même manière

$$\frac{\partial c_j(x^{(k)})}{\partial x_i} = \frac{c_j(x^{(k)} + \delta_{ij} e_i) - c_j(x^{(k)} - \delta_{ij} e_i)}{2\,\delta_{ij}} + o(\delta_{ij}^2) \tag{5.21}$$

permet d'approcher la matrice jacobienne. En toute rigueur, l'évaluation de celle-ci nécessite $2N_xN_c$ simulations du modèle à optimiser[2]. Cependant, les ordres de

[2]Nous supposons que les fonctions $c_j(x)$ ne peuvent être calculées indépendamment l'une de l'autre et que l'évaluation de l'une d'elle nécessite donc une simulation du modèle complet.

grandeurs des fonctions $c_j(x)$ étant généralement comparables, les δ_{ij} peuvent être fixés à une valeur δ_i indépendante de j, ce qui n'entraîne plus que $2N_x$ évaluations.

Il existe également deux formulations similaires, deux fois moins côuteuses en temps de calcul mais moins précises, appelées respectivement différences finies avant et arrière :

$$\frac{\partial f(x^{(k)})}{\partial x_i} = \frac{f(x^{(k)}+\delta_i e_i)-f(x^{(k)})}{\delta_i}+o(\delta_i), \tag{5.22}$$

$$\frac{\partial f(x^{(k)})}{\partial x_i} = \frac{f(x^{(k)})-f(x^{(k)}-\delta_i e_i)}{\delta_i}+o(\delta_i). \tag{5.23}$$

Dans les cas qui nous occupent, quelle que soit la formulation employée, la méthode des différences finies s'avère très coûteuse. De plus, le choix des perturbations δ_i est un problème abondamment traité en analyse numérique [67] et la précision obtenue est insuffisante en cas de non-linéarités importantes. Il est donc, la plupart du temps, tout à fait inenvisageable de procéder de la sorte en raison du nombre important d'itérations à effectuer et du temps de calcul requis pour une simulation.

5.4.2 Différentiation directe.

La différentiation automatique est un moyen de calculer les dérivées d'une fonction par rapport à un ensemble de variables indépendantes. La technique de différentiation directe fonctionne directement sur le code de calcul du modèle direct. Certains logiciels permettent d'effectuer cette opération de façon systématique [86].

Dans le cas qui nous occupe $x_1,\ldots,x_{N_x}$ sont les variables indépendantes et nous nommerons y_i pour $i=1,\ldots,N_y$ les valeurs intermédiaires successivement calculées par le code de calcul. Chaque ligne de code peut ainsi s'écrire de la sorte

$$y_i = F_i(x_1,\ldots,x_{N_x},y_1,\ldots,y_{i-1}) \quad i=1,\ldots,N_y. \tag{5.24}$$

Dans cette méthode directe, pour chaque variable y_i, on peut calculer la valeur correspondante de la dérivée par rapport à la variable x_j. Pour chaque opération (5.24), les dérivées sont calculées par la règle

$$\frac{\partial y_i}{\partial x_j} = \frac{\partial F_i}{\partial x_j} + \sum_{k=1}^{i-1} \frac{\partial F_i}{\partial y_k}\frac{\partial y_k}{\partial x_j} \quad j=1,\ldots,n. \tag{5.25}$$

Notons que, puisque les F_i sont généralement des opérateurs unaires ou binaires[3], la somme de l'équation (5.25) se réduit souvent à un simple terme ou à une somme

[3] Tout code de calcul peut se décomposer en une suite d'opérateurs unaires et binaires.

de deux termes. Cette forme peut aussi être utilisée pour exploiter les possibilités de *surcharge des opérateurs* FORTRAN et ainsi effectuer les dérivations en parallèle du calcul.

Cette méthode permet la génération d'un code qui calcule la dérivée de n'importe quelle quantité par rapport à chaque variable indépendante x_i. Ceci nous permet de calculer les valeurs qui sont exigées par la plupart des méthodes d'optimisation : les composantes du gradient de la fonction objectif $\nabla_x f(x^{(k)})$ au point $x^{(k)}$. La valeur de la matrice jacobienne $J(x^{(k)})$ est également disponible sans coût de calcul supplémentaire.

On remarque que le surcroît de calcul engendré par cette technique est de l'ordre de $2N_x$ fois le coût du modèle direct. En effet, chaque opération du code direct (5.24) entraîne l'apparition de N_x opérations (5.25). Si l'opération de base est unaire, le nombre d'opération ainsi engendré est N_x et le double si l'opérateur est binaire car il faut y ajouter N_x sommations. La majorité des opérations étant binaires, le coût de calcul est donc de l'ordre de la méthode centrée des différences finies et deux fois supérieur à celui des méthodes décentrées mais pour un gain conséquent en précision[4].

5.4.3 Méthode du modèle adjoint.

La méthode du modèle adjoint (aussi appelée méthode de différentiation arrière) nous permet d'écrire des équations qui donnent les composantes du gradient de la fonction objectif au fur et à mesure du calcul (voir par exemple [60, 61, 66, 77, 93, 94, 99]).

Elle se base sur le principe suivant : nous considérons la fonction objectif comme une variable d'état supplémentaire

$$s_{N_s+1} = f(x). \tag{5.26}$$

Notons que la fonction objectif f dépend en fait des paramètres x par l'intermédiaire des valeurs de comparaison c_j qui dépendent elles-mêmes des variables d'état

$$c_j = R_j(s_1, \ldots, s_{N_s}, t) \qquad j = 1, \ldots, m \tag{5.27}$$

et

$$s_\ell(t) = F_\ell(x, t) \tag{5.28}$$

où t désigne le vecteur des variables indépendantes (généralement le temps et l'espace) et les fonctions F_ℓ représentent les solutions des équations du modèle.

[4] La dérivation est exacte, l'erreur est de l'ordre de la précision machine.

Nous pouvons construire le Lagrangien du problème

$$\mathcal{L}(x,s,\lambda) = s_{N_s+1} - \sum_{\ell=1}^{N_s+1} \lambda_\ell \left(s_\ell - F_\ell(x,t)\right) \tag{5.29}$$

où les λ_ℓ sont les variables *duales* ou *adjointes*.

Les conditions de stationnarité du Lagrangien sont

$$\frac{\partial \mathcal{L}}{\partial x_k} = 0 \qquad k = 1,\dots,n \tag{5.30}$$

$$\frac{\partial \mathcal{L}}{\partial s_k} = 0 \qquad k = 1,\dots,N_s+1 \tag{5.31}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda_k} = 0 \qquad k = 1,\dots,N_s+1. \tag{5.32}$$

Connaissant l'expression du Lagrangien (5.29), nous pouvons évaluer (5.32) et nous obtenons simplement les équations (5.28) pour $k = 1,\dots,N_s$ et (5.26) pour $k = N_s+1$.

De même, les relations (5.31) nous donnent

$$\frac{\partial \mathcal{L}}{\partial s_k} = 0 = \frac{\partial s_{N_s+1}}{\partial s_k} - \lambda_k \tag{5.33}$$

ce qui nous permet de déduire les variables adjointes

$$\lambda_k = \frac{\partial s_{N_s+1}}{\partial s_k} = \frac{\partial f}{\partial s_k} \quad \text{pour } k = 1,\dots,N_s \tag{5.34}$$

et

$$\lambda_{N_s+1} = 1 \quad \text{pour } k = N_s+1. \tag{5.35}$$

Le Lagrangien se réécrit donc

$$\mathcal{L}(x,s,\lambda) = f(x) - \sum_{\ell=1}^{N_s} \lambda_m \left(s_\ell - F_\ell(x,t)\right) \tag{5.36}$$

que nous injectons enfin dans les expressions (5.30) pour obtenir

$$\frac{\partial \mathcal{L}}{\partial x_k} = 0 = \frac{\partial f}{\partial x_k} - \sum_{\ell=1}^{N_s} \lambda_\ell \frac{\partial F_\ell}{\partial x_k} \tag{5.37}$$

ce qui nous permet de calculer les composantes du gradient de la fonction objectif par rapport aux variables de contrôle :

$$\frac{\partial f}{\partial x_k} = \sum_{\ell=1}^{N_s} \lambda_\ell \frac{\partial F_\ell}{\partial x_k} \quad \text{pour } k = 1,\dots,n. \tag{5.38}$$

En résumé, le raisonnement à suivre pour obtenir le gradient de la fonction objectif est le suivant :

– nous devons résoudre les équations du modèle pour obtenir les expressions (5.28) ;
– nous calculons les variables adjointes par les équations adjointes (5.34) ;
– connaissant les variables adjointes, nous pouvons obtenir les composantes du gradient de la fonction objectif en dérivant (5.28) et en utilisant la formule (5.38).

En pratique, nous ne disposons évidemment des expressions (5.28) que par voie numérique, nous ne pouvons donc formellement dériver celles-ci pour calculer les variables adjointes de Lagrange des variables de contrôle.

Les équations du modèle discrétisées s'écrivent sous la forme (5.24)

$$y_i = F_i(x_1, \ldots, x_n, y_1, \ldots, y_{i-1}) \quad i = 1, \ldots, N_y \tag{5.39}$$

dans laquelle nous tenons compte du fait que ces variables sont calculées dans un ordre bien précis. Considérons que la fonction objectif est la dernière variable calculée dans le modèle numérique direct

$$f(x) = y_{N_y} = F_{N_y}(x_1, \ldots, x_n, y_1, \ldots, y_{N_y-1}). \tag{5.40}$$

Le code numérique est une succession d'équations de type (5.24), la méthode des multiplicateurs de Lagrange nous suggère de construire le Lagrangien pour chaque étape du calcul, c'est-à-dire pour chaque ligne de code. Toute ligne de code du modèle générera une ou plusieurs lignes de code dans le programme de résolution des équations adjointes[5]. Soit les lignes du code direct

$$\begin{array}{rcl} Y & = & G(X, \ldots) \\ Z & = & F(X, Y, \ldots) \end{array}$$

où Y est une variable intermédiaire. L'ordre alphabétique indique l'ordre dans lequel les opérations ont été effectuées (X d'abord, puis Y, puis Z). Les contributions au Lagrangien de ces lignes de code seront

$$\mathcal{L} = \ldots - \lambda_Y \{Y - G(X, \ldots)\} - \lambda_Z \{Z - F(X, Y, \ldots)\} + \ldots. \tag{5.41}$$

Des variables adjointes doivent dès lors être introduits pour chaque variable apparaissant aux membres de droite des lignes du code direct. La stationnarité du Lagrangien s'écrit sous la forme

$$\frac{\partial \mathcal{L}}{\partial Y} = -\lambda_Y + \lambda_Z \frac{\partial F}{\partial Y} = 0 \tag{5.42}$$

[5] Si nous considérons que chaque ligne de code peut être décomposée en une suite d'opérateurs unaires ou binaires, il en résultera respectivement une ou deux opérations dans le code adjoint.

ce qui nous permet de calculer la variable adjointe

$$\lambda_Y = \lambda_Z \frac{\partial F}{\partial Y} \tag{5.43}$$

si λ_Z est connu. Nous constatons à nouveau que l'implémentation des équations adjointes doit s'effectuer dans l'ordre inverse des opérations pour les équations du modèle direct (λ_Z, puis λ_Y, puis λ_X).

Pour une variable Y quelconque, chaque ligne du code direct de la forme $Z = F(X,Y,\ldots)$ induit une contribution à la variable adjointe de Y d'une quantité

$$\lambda_Z \frac{\partial F}{\partial Y}. \tag{5.44}$$

Puisque ces termes doivent être accumulés pour chaque équation où Y apparaît dans le membre de droite, la ligne de code qui apparaît dans le modèle adjoint est

$$\lambda_Y := \lambda_Y + \lambda_Z \frac{\partial F}{\partial Y} \tag{5.45}$$

en ayant bien pris soin d'initialiser λ_Y à zéro.

Les paramètres x_i du modèle ne figurent jamais dans le membre de gauche et apparaissent forcément au membre de droite d'au moins une ligne d'instruction dans le code direct, ce qui entraîne la création d'une variable adjointe s'y rapportant dans le code adjoint. A la fin du code adjoint ceux-ci donnent les valeurs des composantes du gradient de la fonction objectif par rapport aux variables de contrôle

$$\frac{\partial f}{\partial x_k} = \lambda_{x_k} \qquad k = 1,\ldots,n. \tag{5.46}$$

Le coût numérique de cette méthode est de l'ordre de deux fois celui du modèle direct. En effet, chaque opération élémentaire (unaire ou binaire) résultera en un maximum de deux opérations adjointes. Ceci en fait une méthode de prédilection lorsque le nombre de paramètres à optimiser devient important. L'inconvénient de cette méthode est la nécessité de garder en mémoire nombre de variables d'état, ce qui peut exiger des ressources considérables dans certains cas. Un second désavantage est son incapacité à donner d'autres informations que le gradient $\nabla_x f$. Les méthodes d'optimisation requérant l'évaluation de la matrice jacobienne sont donc nécessairement à écarter aussitôt que cette technique de différentiation est employée.

5.4.4 Coût en ressources informatiques.

Le coût d'un code de calcul a deux sources : la mémoire nécessaire et le nombre d'opérations à effectuer. Idéalement ces deux-ci doivent être réduits au

maximum ; il n'est pas rare d'avoir un seul de ces deux facteurs qui conditionne complètement le coût informatique global. Généralement, ce qui peut être gagné d'un côté est perdu de l'autre. La différentiation numérique ne déroge pas à cette règle. Nous allons procéder ici à une comparaison intuitive des méthodes de différentiation et de certaines variantes (voir [101]).

Le premier point de comparaison, déjà évoqué, est l'information retirée. Là où la méthode de différentiation directe permet une évaluation aussi bien du gradient $\nabla_x f$ que de la matrice jacobienne J, le modèle adjoint ne donne accès qu'au gradient. La méthode directe s'impose donc aussitôt que l'on désire évaluer la matrice jacobienne.

Le second point de comparaison est le nombre d'opérations nécessaires. Celui-ci est plus important pour la méthode de différentiation directe ; le gain d'information a donc un prix. Pour formaliser la suite de la discussion, nous définirons C_m, C_d et C_a, respectivement le nombre d'opérations[6] nécessaires au calcul du modèle lui-même, du code de différentiation directe et du modèle adjoint. Comme estimé précédemment, nous avons les ordres de grandeur suivants

$$\begin{aligned} C_d &= O(2nC_m) \\ C_a &= O(2C_m). \end{aligned}$$

Ce ne sont là que des ordres de grandeur, un programmeur habile pourra faire passer le coût en deçà de celui prédit.

Un troisième critère de performance est l'utilisation de la mémoire. La méthode ajointe est la plus coûteuse de ce point de vue : elle nécessite le stockage de toutes les variables d'état du modèle direct[7], ce qui peut s'avérer considérable. Néanmoins une technique, dite du « checkpointing » permet de réduire l'utilisation de mémoire. Cette technique consiste à ne stocker qu'un nombre réduit de valeurs et à recalculer les autres en cours de simulation du modèle adjoint. Au final, le nombre d'opérations est accru d'une simulation entière du modèle direct qui permet de réduire la consommation de mémoire. La méthode de différentiation directe ne souffre, quant à elle, pas de ce désagrément. En effet les opérations du code de dérivation peuvent s'effectuer de pair avec la simulation du modèle direct, les variables d'état nécessaires au calcul des dérivées sont donc disponibles en cours de calcul.

Enfin, un quatrième angle de comparaison apparaît si une simulation du modèle direct n'implique pas toujours une évaluation des dérivées. Il peut arriver, et ce sera le cas dans les développements qui suivent, qu'il soit intéressant de « découpler les deux codes de calcul ». Qu'entendons nous par « découplage des deux

[6]Celui-ci sera souvent assimilé, par abus de langage, au temps de calcul.

[7]Ce qui signifie pour un modèle discrétisé sur le temps et l'espace, le stockage de toutes les valeurs des variables d'état pour tous les pas de temps et pour toutes les mailles.

TAB. 5.1 – Tableau comparatif des différentes méthodes de différentiation.

Méthode	Précision	Information disponible	Temps de calcul	Utilisation mémoire	Découplage
Diff. directe de base	Précision machine	Jacobienne et gradient	$2nC_m$	Faible	Non
Diff. directe à mémoire	Précision machine	Jacobienne et gradient	$2nC_m$	Importante	Oui
Diff. directe avec resimulation	Précision machine	Jacobienne et gradient	$(2n+1)C_m$	Faible	Oui
Modèle adjoint	Précision machine	Gradient uniquement	$2C_m$	Importante	Oui
Modèle adjoint checkpointed	Précision machine	Gradient uniquement	$3C_m$	Moyenne	Oui
Différences finies centrées	$o(\delta^2)$	Jacobienne et gradient	$2nC_m$	Très faible	Oui
Différences finies avant ou arrière	$o(\delta)$	Jacobienne et gradient	nC_m	Très faible	Oui

codes de calcul » ? Nous dirons que les codes sont découplés si nous pouvons exécuter le calcul de la fonction objectif (et donc une simulation du modèle direct) sans qu'il soit nécessaire de spécifier *a priori* si l'évaluation des dérivées est nécessaire. La méthode adjointe, que ce soit dans sa version originale ou « checkpointed », est découplée. En effet, une simulation du modèle direct permet de calculer la fonction objectif et les variables d'état sont gardées en mémoire. Si une évaluation des dérivées est nécessaire, il suffit d'opérer à la simulation du modèle adjoint et dans le cas contraire d'effacer le contenu de la mémoire. La méthode directe dans sa version originale ne possède pas cette propriété : l'utilisateur doit spécifier avant la simulation s'il désire voir s'effectuer le calcul des dérivées ou non. Deux techniques permettent de découpler la méthode. La première, que nous nommerons différentiation directe à mémoire, consiste à conserver les variables d'état en mémoire pour les réutiliser ensuite s'il s'avérait nécessaire de calculer les dérivées du modèle. Ceci engendre naturellement une utilisation de mémoire équivalente à la méthode adjointe. La seconde est une lapalissade : il suffit simplement d'effectuer une simulation du modèle, et d'en effectuer une seconde au cas où le calcul des dérivées est exigé. Nous nommerons ce dernier procédé différentiation directe avec resimulation. Les différentes caractéristiques des méthodes sont résumées dans le tableau 5.1.

5.4.5 Exemple : l'oscillateur harmonique.

Soit un oscillateur harmonique dont l'écart y par rapport à sa position de repos répond à l'équation

$$\frac{d^2y}{dt^2} + \omega^2 y = 0 \tag{5.47}$$

avec les conditions initiales

$$y(0) = A \quad \text{et} \quad \frac{dy}{dt}(0) = 0 \tag{5.48}$$

dont la solution est

$$y(t) = A\cos(\omega t). \tag{5.49}$$

Si nous utilisons un schéma d'Euler centré avec un pas h pour résoudre numériquement l'équation différentielle, nous obtenons les lignes de code suivantes[8] dans l'implémentation du modèle direct

```
y(0)=A
f=0.5*(y(0)-ytilde(0))**2

y(1)=A-0.5*(omega*h)**2*A
f=f+0.5*(y(1)-ytilde(1))**2

do i=2,N
    y(i)=(2-(omega*h)**2)*y(i-1)-y(i-2)
    f=f+0.5*(y(i)-ytilde(i))**2
enddo
```

5.4.5.1 Différentiation directe.

Si nous désignons par les préfixe « `A_` » et « `omega_` », les dérivées d'une variable donnée respectivement par rapport à A et ω, nous obtenons, en appliquant systématiquement la relation (5.25), les lignes de code suivantes pour le calcul des dérivées

```
A_y(0)=1.
omega_y(0)=0.
A_f=(y(0)-ytilde(0))*A_y(0)
omega_f=(y(0)-ytilde(0))*omega_y(0)

A_y(1)=1-0.5*(omega*h)**2
```

[8] En langage FORTRAN, le symbole « `**` » indique une exponentiation.

```
omega_y(1)=-A*omega*h**2
A_f=A_f+(y(1)-ytilde(1))*A_y(1)
omega_f=omega_f+(y(1)-ytilde(1))*omega_y(1)

do i=2,N
   A_y(i)=(2-(omega*h)**2)*A_y(i-1)-A_y(i-2)
   omega_y(i)=(2-(omega*h)**2)*omega_y(i-1)-omega_y(i-2)&
        &-2*y(i-1)*omega*h**2
   omega_f=omega_f+(y(i)-ytilde(i))*omega_y(i)
   A_f=A_f+(y(i)-ytilde(i))*A_y(i)
enddo
```

On constate que le nombre d'instructions a été doublé, étant donné qu'il y a deux variables de contrôle A et ω.

5.4.5.2 Méthode du modèle adjoint.

Si nous désignons les variables adjointes d'une variable donnée en ajoutant simplement le préfixe « ad_ » à la variable directe correspondante, nous obtenons, par application systématique de la formule (5.45),

```
ad_f=1
do i=N,2,-1
    ad_y(i)=ad_y(i)+(y(i)-ytilde(i))*ad_f
    ad_y(i-1)=ad_y(i-1)+(2-(omega*h)**2)*ad_y(i)
    ad_y(i-2)=ad_y(i-2)-ad_y(i)
    ad_omega=ad_omega-2*omega*y(i-1)*h**2*ad_y(i)
enddo

ad_y(1)=ad_y(1)+(y(1)-ytilde(1))*ad_f
ad_y(0)=ad_y(0)+(y(0)-ytilde(0))*ad_f

ad_omega=ad_omega-omega*A*h**2*ad_y(1)
ad_A=ad_A+(1-0.5*(omega*h)**2)*ad_y(1)
ad_A=ad_A+ad_y(0)
```

en n'oubliant pas d'initialiser toutes les variables adjointes à zéro. Les composantes du gradient de la fonction de coût sont ad_A et ad_omega ; ces valeurs sont exactes aux erreurs d'arrondis près.

5.4.6 Problèmes connus.

Les techniques de différentiation automatique doivent être utilisées avec prudence. Les procédés décrits ici tombent en effet facilement dans des chausse-trappes communs s'ils sont appliqués sans esprit critique. Illustrons notre propos par un petit exemple qui permet de mesurer l'importance d'une implémentation soigneuse du modèle direct.

Exemple 5.1 *Considérons la fonction*

$$f(x) = x^2 - 1 \tag{5.50}$$

dont le code

```
if (x.eq.1.0) then
    f=0.0
else
    f=x**2-1.0
end if
```

est une implémentation en FORTRAN *correcte bien qu'inhabituelle. Dans ce cas, un logiciel de différentiation risque de dériver les deux branches séparément : la dérivée obtenue sera* 0 *si* $x = 1$ *et* $2x$ *dans le cas contraire.*

La différentiation automatique est un sujet de recherche très actif et des compilateurs ou des programmes sont désormais facilement accessibles ; citons, par exemple, ADIFOR [3], ADOL-C [48], AD01 [86], TAPENADE [49], TAF [38] et Odyssée [27]. Ces systèmes de différentiation automatique ont fait leurs preuves sur des problèmes parfois très difficiles et de grande taille. Néanmoins, ces procédés ne doivent pas être considérés comme une arme ultime pour s'affranchir de réfléchir au calcul des dérivées. En effet, de nombreux problèmes gênants subsistent comme celui de la gestion des *pointeurs*, des *allocations dynamiques*, des programmes *parallélisés* ou encore, plus simplement, des *embranchements*.

5.5 Conclusion.

L'identification paramétrique est une étape clé dans le développement d'un modèle mathématique. Celle-ci doit toujours avoir lieu si l'on exige du modèle des résultats précis. Pour les modèles les plus sophistiqués, une systématique dans la résolution de ce problème est la bienvenue. Des concepts tels que les expériences jumelles ou les analyses post-optimisation sont de précieux outils à cet égard.

Les problèmes d'identification paramétrique sont parmi les problèmes d'optimisation non-contrainte les plus répandus et les méthodes de calcul utilisées requièrent généralement l'évaluation du gradient de la fonction objectif ou de la matrice jacobienne. Ceux-ci peuvent être obtenus avec une excellente précision en utilisant la méthode de différentiation directe ou un modèle adjoint. Cependant, ces dernières exigent des ressources informatiques différentes et celles-ci doivent être prises en compte dans l'évaluation du coût de la méthode d'optimisation correspondante.

Chapitre 6

Trust : un algorithme d'optimisation par régions de confiance

Nous nous attachons, dans ce chapitre, à décrire un algorithme d'optimisation par régions de confiance spécifiquement développé pour s'attaquer aux problèmes d'identification paramétrique introduits au chapitre précédent. Plusieurs variantes sont étudiées et comparées. L'algorithme se base sur le schéma général décrit au chapitre 4. Il utilise des approximations quadratiques et prend en compte des contraintes de bornes.

Dans le cadre de ce travail, un algorithme nommé « Trust » a été implémenté dans une routine FORTRAN. Il s'agit d'un algorithme utilisant une globalisation par régions de confiance et des approximations globales quadratiques. Plusieurs versions sont développées utilisant des approximations locales de type quasi-Newton ainsi qu'une version utilisant l'approximation locale de Gauss–Newton. Ces différentes méthodes ont été utilisées, avec succès, dans divers travaux d'identification paramétrique [51, 52, 100, 101]. Trust a été conçue pour résoudre un problème de la forme

$$\text{minimiser} \quad f(x) \tag{6.1}$$

$$\text{s.c.} \quad x_L \leq x \leq x_U. \tag{6.2}$$

dans laquelle $f(x)$ est supposée satisfaire aux hypothèses 4.1, 4.2 et 4.3. Les vecteurs x_L et $x_U \in \mathbb{R}^n$ sont des contraintes de bornes. (Bien qu'absentes de la théorie développée dans le chapitre précédent, la possibilité d'adjoindre des contraintes de bornes a été ajoutée en raison de leur fréquence dans le domaine de l'identification paramétrique pour lequel Trust a été développé au départ.) La routine implémentée inclut également des fonctionnalités pour faciliter la mise à échelle et l'impression personnalisée.

Une application est ensuite présentée. Il s'agit d'une identification paramétrique d'un système dynamique simple : le modèle de Lotka–Volterra. Les perfor-

mances de différentes versions de Trust sont analysées et comparées à la routine M1QN3 de Gilbert et Lemaréchal [40]. Le mode d'emploi complet est présenté en annexe A.

6.1 Sous-problèmes quadratiques.

L'algorithme implémenté fait usage d'approximations locales quadratiques (voir section 3.2), *i.e.* de la forme

$$m^{(k)}(x^{(k)}+s) = f(x^{(k)}) + s^T g^{(k)} + \frac{1}{2} s^T H^{(k)} s \tag{6.3}$$

avec

$$g^{(k)} = \nabla_x f(x^{(k)}). \tag{6.4}$$

Nous supposons que l'utilisateur peut fournir les valeurs de la fonction objectif et de son gradient en un point donné (et éventuellement de la matrice jacobienne si c'est un problème de moindres carrés). La norme utilisée pour définir la région de confiance est la norme euclidienne et n'est pas fonction de l'itération. La région

$$\mathcal{B}^{(k)} = \left\{ x \in \mathbb{R}^n : \|x - x^{(k)}\| \leq \Delta^{(k)} \right\} \tag{6.5}$$

est donc une hyper-sphère de rayon $\Delta^{(k)}$.

Nous utiliserons plusieurs expressions différentes pour la mise à jour du Hessien de l'approximation locale qui constitueront autant de versions de l'algorithme.

Trust-SR1 conditionnelle : La mise à jour de la matrice hessienne $H^{(k)}$ se fait par une technique de type quasi-Newton. La mise à jour symétrique de rang un (3.46) n'est cependant effectuée que si l'itération est réussie. Dans ce cas, le point $\hat{x}^{(k-1)}$ utilisé dans (3.41) pour construire l'approximation quadratique est simplement l'itéré précédent $x^{(k-1)}$. Les définitions de $r^{(k)}$ et $y^{(k)}$, respectivement (3.41) et (3.42), deviennent donc

$$r^{(k)} = x^{(k)} - x^{(k-1)} \tag{6.6}$$

$$y^{(k)} = g^{(k)} - g^{(k-1)} \tag{6.7}$$

Notons qu'en cas d'itération infructueuse, il n'est pas nécessaire d'évaluer le gradient de la fonction objectif au point de test $\tilde{x}^{(k)}$. Si le calcul de la fonction objectif et de son gradient sont découplés, il est donc possible d'épargner le coût numérique associé au calcul du gradient.

Trust-BFGS conditionnelle : Cette version est en tout point semblable à la précédente à la seule exception près qu'elle utilise la mise à jour BFGS (3.57) en lieu et place de (3.46).

Trust-SR1 inconditionnelle : Tout comme dans la version « conditionnelle », la mise à jour de la matrice hessienne $H^{(k)}$ se fait par une technique de type quasi-Newton, utilisant la formule symétrique de rang un (3.46) à chaque itération. Le point $\hat{x}^{(k-1)}$ utilisé pour construire l'approximation quadratique est simplement l'itéré précédent $x^{(k-1)}$ si l'itération est réussie et le point de test $\tilde{x}^{(k)}$ si celle-ci s'avère infructueuse, ce qui implique — contrairement à ce qui se passe dans l'approche conditionnelle — d'y évaluer le gradient de la fonction objectif.

Trust-BFGS inconditionnelle : Cette version est en tout point semblable à la précédente à la seule exception près qu'elle utilise la mise à jour BFGS (3.57) en lieu et place de (3.46).

Trust-GN : Cette version est utilisée pour les problèmes de *moindres carrés* pour lesquels la fonction objectif s'écrit sous la forme (3.67). Trust-GN s'inspire de la méthode de Gauss-Newton (3.72) pour l'approximation du Hessien. Contrairement aux méthodes précédentes, celle-ci nécessite, en plus de l'évaluation du gradient, celle de la matrice jacobienne $G(x^{(k)})$. Notons que l'approximation locale utilisée ne dépend que des valeurs du gradient et de la matrice jacobienne à l'itéré $x^{(k)}$. L'évaluation du gradient et de la matrice jacobienne n'est dès lors nécessaire que pour les itérations réussies.

À chaque itération, il convient de résoudre un sous-problème quadratique soumis à une contrainte de confinement dans une région de confiance sphérique. Il s'agit donc de trouver le pas de progression s_M de façon à résoudre le problème[1]

$$\text{minimiser} \quad q(s) = g^T s + \frac{1}{2} s^T H s \tag{6.9}$$

$$\text{s.c.} \quad \|s\| \leq \Delta \tag{6.10}$$

avec $s \in \mathbb{R}^n$.

Certaines propriétés apparaissent immédiatement si nous analysons ce sous-problème. La solution que nous cherchons est soit à l'intérieur de la région de confiance, c'est-à-dire telle que $\|s\| < \Delta$, soit sur la frontière auquel cas $\|s\| = \Delta$. Si la solution est intérieure, la contrainte de confinement est inactive et s_M est un

[1] Dans un souci de simplicité, nous avons supprimé le compteur d'itérations (k) et défini

$$q(s) = m^{(k)}(x^{(k)} + s) - m^{(k)}(x^{(k)}). \tag{6.8}$$

minimum[2] non-contraint de $q(s)$ et le multiplicateur de Lagrange μ_M relatif à la contrainte (6.10) est nul. Notons que ceci ne peut se produire que dans le cas où $q(s)$ est convexe, c'est-à-dire que le Hessien est semi-défini positif. Nous voyons donc que le comportement sera différent suivant que l'approximation locale est convexe ou non. Dans ce dernier cas, la solution est nécessairement sur la frontière de la région de confiance, celle-ci est active et μ_M est positif (ou éventuellement nul). Par contre, dans le cas où $q(s)$ est convexe, la solution peut aussi bien se situer à l'intérieur de la région de confiance que sur sa frontière. La solution s_M peut être caractérisée plus finement par le théorème suivant.

Théorème 6.1 *Le vecteur s_M est un minimum global du sous-problème* (6.9) *soumis à la contrainte* (6.10) *si et seulement si $\|s_M\| \leq \Delta$ et s'il existe un multiplicateur de Lagrange scalaire μ tel que*

$$(H + \mu_M I)s_M = -g \quad (6.11)$$

$$\mu_M(\Delta - \|s_M\|) = 0 \quad (6.12)$$

et $(H + \mu_M I)$ est symétrique semi-définie positive.

La preuve de ce théorème peut être trouvée dans [74, 80, 20]. Partant de ce théorème, Moré et Sorensen [74] ont développé un algorithme qui est résumé ci-après. Ces mêmes auteurs l'ont également implémenté et mis à disposition dans la routine GQT qui est utilisée dans ce travail.

6.1.1 Méthode de Moré et Sorensen.

Nocedal et Wright [80] décrivent sommairement l'algorithme de la façon suivante. Si la matrice H est symétrique définie positive et si le minimum non-contraint correspondant se situe à l'intérieur de la région de confiance, il est bel et bien le minimum global. Dès lors, si H est symétrique définie positive, son inversion est effectuée au moyen d'une *décomposition de Cholesky* qui permet de la factoriser en un produit $R^T R$ où R est une matrice triangulaire supérieure. Le minimum non-contraint de l'approximation locale s'obtient alors en résolvant successivement deux systèmes triangulaires. Le minimum global s_M est égal au minimum non-contraint si celui-ci est à l'intérieur de la région de confiance, *i.e.* si $\|s\| \leq \Delta$.

Dans les cas contraires, nous définissons

$$s(\mu) = -(H + \mu I)^{-1} g, \quad (6.13)$$

[2] Ce minimum est nécessairement global vu la forme de $q(s)$.

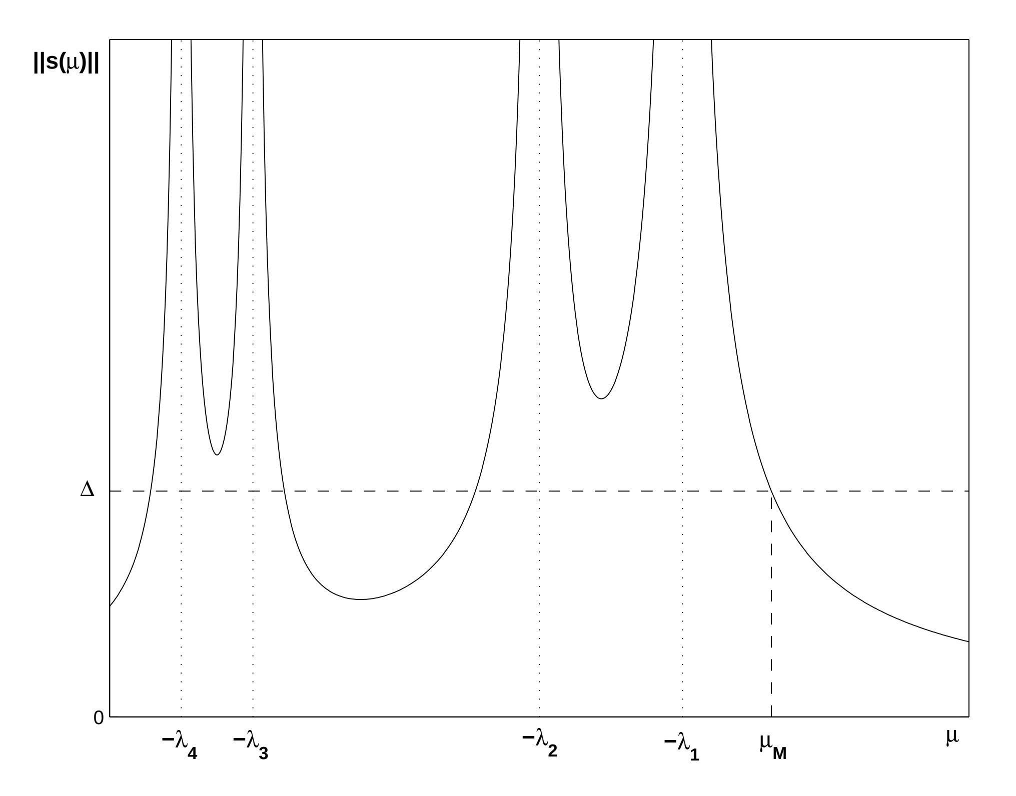


FIG. 6.1 – Profil de la fonction $\|s(\mu)\|$ lorsque $v_1^T g \neq 0$.

solution de (6.11) pour des valeurs de μ suffisamment grandes pour que $H + \mu I$ soit symétrique définie positive. La décomposition spectrale de l'inverse de cette matrice permet d'écrire

$$s(\mu) = -\sum_{j=1}^{n} \frac{v_j^T g}{\lambda_j + \mu} v_j, \tag{6.14}$$

où λ_j est une valeur propre de H et v_j le vecteur propre correspondant. Vu la symétrie de H, les valeurs propres λ_j sont réelles et les vecteurs propres v_j peuvent être choisis orthogonaux. La norme du vecteur $s(\mu)$ se calcule aisément en tenant compte de cette orthogonalité

$$\|s(\mu)\| = \sqrt{s(\mu)^T s(\mu)} = \sqrt{\sum_{j=1}^{n} \left(\frac{v_j^T g}{\lambda_j + \mu} \right)^2}. \tag{6.15}$$

Sans perte de généralité, nous pouvons supposer $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

Nous savons que la contrainte (6.10) est active et nous obtenons dès lors une

équation unidimensionnelle pour la variable μ

$$\|s(\mu)\| = \Delta \tag{6.16}$$

dont la solution est μ_M. Si $\mu > -\lambda_1$, les dénominateurs intervenant dans (6.15) sont strictement positifs et

$$\lim_{\mu \to +\infty} \|s(\mu)\| = 0. \tag{6.17}$$

De plus, si $v_1^T g \neq 0$, nous avons

$$\lim_{\mu \to -\lambda_1} \|s(\mu)\| = +\infty \tag{6.18}$$

et il existe dès lors une seule valeur $\mu_M \in]-\lambda_1, +\infty[$ pour laquelle $\|s(\mu_M)\| = \Delta$ (voir figure 6.1). Sachant cela, l'équation (6.16) peut être résolue avec un algorithme unidimensionnel classique de recherche de racine. Toutefois, cette équation s'avère généralement mal conditionnée et on lui préfère la forme équivalente

$$\frac{1}{\|s(\mu)\|} = \frac{1}{\Delta}. \tag{6.19}$$

Une fois μ_M calculé, s_M est directement obtenu à partir de (6.11) où la matrice $H + \mu_M I$ est symétrique définie positive.

6.1.2 Le « hard case » de Moré et Sorensen.

Dans la discussion ci-dessus, nous avons occulté le cas où $v_1^T g = 0$. Notons que cette discussion reste néanmoins valable si la plus petite valeur propre est une valeur propre multiple (*i.e.* $\lambda_1 = \lambda_2 = \ldots$) pour autant que $v_j^T g \neq 0$ pour au moins un des vecteurs propres relatifs à λ_1. Dans les cas contraires, la limite (6.17) prend une valeur finie δ. Si $\delta < \Delta$, cela signifie qu'aucune valeur de $\mu \in]-\lambda_1, +\infty[$ n'est telle que $\|s(\mu)\| = \Delta$.

C'est ce cas de figure que Moré et Sorensen [74] nomment le « hard case ». À première vue, il est difficile de choisir s_M et μ_M qui satisferaient aux conditions énoncées par la théorème 6.1 qui nous garantit cependant que ce dernier existe dans l'intervalle $[-\lambda_1, +\infty[$. Il n'y a donc qu'une seule possibilité : $\mu_M = -\lambda_1$. Dans ce cas, la matrice $H - \lambda_1 I$ est cependant singulière et la décomposition spectrale de son inverse ne peut plus être utilisée telle quelle pour définir (6.14). Il convient de définir le vecteur

$$s(\tau) = -\sum_{j:\lambda_j \neq \lambda_1} \frac{v_j^T g}{\lambda_j - \lambda_1} v_j + \tau v_1 \tag{6.20}$$

dont on vérifie aisément qu'il remplit la condition (6.11). En tenant compte des relations d'orthogonalité entre les vecteurs v_j, nous obtenons

$$\|s(\tau)\| = \sqrt{\sum_{j:\lambda_j \neq \lambda_1} \left(\frac{v_j^T g}{\lambda_j - \lambda_1} \right)^2 + \tau^2} = \sqrt{\delta^2 + \tau^2} \tag{6.21}$$

et il est dès lors toujours possible de choisir τ tel que $\|s(\tau)\| = \Delta$. On vérifie aisément que le vecteur ainsi trouvé est bel et bien le minimum s_M du problème (6.9) soumis à la contrainte (6.10).

6.2 Aspects pratiques de l'implémentation.

Après le type d'approximation locale et la forme de la région de confiance utilisés ainsi que la méthode de résolution du sous-problème, il reste encore quelques degrés de liberté à fixer dans l'algorithme 2.1.

6.2.1 Mise à jour du rayon de confiance.

Le premier d'entre eux est la stratégie de mise à jour du rayon de confiance. Nous avons opéré le choix suivant

$$\Delta^{(k+1)} = \begin{cases} \alpha_1 \Delta^{(k)} & \text{si } \rho^{(k)} < \eta_1, \\ \Delta^{(k)} & \text{si } \eta_1 \leq \rho^{(k)} < \eta_2, \\ \alpha_2 \Delta^{(k)} & \text{si } \eta_2 \leq \rho^{(k)} \leq \eta_3, \\ \alpha_3 \Delta^{(k)} & \text{si } \rho^{(k)} > \eta_3 \end{cases} \tag{6.22}$$

avec

$$0 < \eta_1 \leq \eta_2 < 1 < \eta_3 \tag{6.23}$$

et

$$\alpha_1 < 1 < \alpha_3 < \alpha_2. \tag{6.24}$$

Cette mise à jour entre bien dans le cadre général (2.30) et dans celui, légèrement plus restrictif, établi par (4.38). Pour s'en convaincre, il suffit de prendre $\gamma_1 = \gamma_2 = \alpha_1$, $\gamma_3 = \alpha_3$ et $\gamma_4 = \alpha_2$. Les valeurs effectivement utilisées lors des expériences numériques sont, sauf mention contraire, celles portées au tableau 6.1. La question de la mise à jour du rayon de confiance est traitée de manière plus approfondie au chapitre 7.

TAB. 6.1 – Valeur numérique des paramètres de la stratégie de mise à jour du rayon de confiance utilisées dans les expériences numériques.

Paramètre	Valeur	Paramètre	Valeur
α_1	$0,5$	η_1	$0,01$
α_2	2	η_2	$0,95$
α_3	$1,01$	η_3	$1,05$

6.2.2 Calcul du rapport $\rho^{(k)}$.

D'après Conn *et al.* [20], une des phases les plus dangereuses dans une méthode par régions de confiance s'avère — de façon plutôt surprenante — être celle où la suite d'itérés s'apprête à atteindre le point critique vers lequel elle converge. Dans ce cas, le numérateur et le dénominateur du rapport (2.29)

$$\rho^{(k)} = \frac{f(x^{(k)}) - f(\tilde{x}^{(k)})}{m^{(k)}(x^{(k)}) - m^{(k)}(\tilde{x}^{(k)})} \tag{6.25}$$

seront petits et le calcul peut souffrir des effets de l'arithmétique en virgule flottante.

En pratique, pour une valeur $\varsigma > 0$ de l'ordre de dix fois la précision machine, nous calculons

$$\delta f^{(k)} = f(\tilde{x}^{(k)}) - f(x^{(k)}) - \varsigma \max(1, |f(x^{(k)})|) \tag{6.26}$$

et

$$\delta m^{(k)} = m^{(k)}(\tilde{x}^{(k)}) - m^{(k)}(x^{(k)}) - \varsigma \max(1, |f(x^{(k)})|) \tag{6.27}$$

puis utilisons la valeur

$$\rho^{(k)} = \begin{cases} 1 & \text{si } \delta f^{(k)} < \varsigma \text{ et } |f(x^{(k)})| > \varsigma, \\ \delta f^{(k)} / \delta m^{(k)} & \text{sinon.} \end{cases} \tag{6.28}$$

6.2.3 Mise à échelle.

Comme évoqué dans la section 4.4, il est assez fréquent que, dans un problème pratique, les variables aient des ordres de grandeurs sensiblement différents. Dans ce cas, il s'avère intéressant de travailler avec des régions de confiance elliptiques, *i.e.* définies par (4.43) et (4.46). Le cas échéant, nous travaillerons donc avec des variables normalisées

$$\xi_i = \frac{x_i - x_i^{\text{car}}}{\delta x_i^{\text{car}}} \tag{6.29}$$

où x_i^{car} est une valeur caractéristique et δx_i^{car} une échelle de variation caractéristique.

Il est en effet évident que, dans l'espace des x_i, la région de confiance idéale serait un ellipsoïde d'autant plus étendu dans une direction que l'échelle de variation de la variable correspondante est grande. Le changement de variable (6.29) nous permet d'utiliser simplement une sphère dans l'espace des ξ_i. Notons que, si des variations caractéristiques sont données, la valeur initiale pour le rayon de confiance $\Delta^{(0)} = 1$ s'impose pratiquement à nous puisqu'une variation de la variable ξ_i de l'ordre de l'unité provoquera une variation de x_i de l'ordre de δx_i^{car}.

6.2.4 Contraintes de bornes.

La plupart des problèmes non-contraints, particulièrement lorsqu'il s'agit de problèmes d'identification paramétrique, font intervenir des contraintes dites « de bornes », *i.e.* pour $i = 1, \ldots, n$

$$\underline{x}_i \leq x_i \leq \overline{x}_i \tag{6.30}$$

où $\underline{x}_i$ et $\overline{x}_i$ sont respectivement les bornes inférieure et supérieure de la variable x_i. La variable x_i est dite fixe et ne constitue plus une variable pour l'optimisation si $\underline{x}_i = \overline{x}_i$.

Ce type de contrainte est plutôt simple à traiter en raison de son caractère linéaire. Il s'avère donc utile, dans un algorithme comme Trust, d'implémenter une stratégie de contraintes actives. La stratégie adoptée est classique. À une itération donnée k, nous définissons tout d'abord l'ensemble $\mathcal{F}^{(k)}$ des indices des variables *fixées* à une de leurs bornes. Dans la même logique, nous définissons l'ensemble des indices des variables *fixées au départ*[3]

$$\mathcal{F}^{(0)} = \{i : \underline{x}_i = \overline{x}_i\}. \tag{6.31}$$

Nous définissons également l'espace vectoriel $\mathcal{A}^{(k)}$ des variables actives à l'itération k comme le sous-espace de $\mathbb{R}^n$ tel que les variables fixées — *i.e.* les variables x_i pour lesquelles $i \in \mathcal{F}^{(k)}$ — soient égales à la borne (inférieure ou supérieure) sur laquelle elles ont été fixées.

À chaque itération, la minimisation de l'approximation locale quadratique s'effectue non pas sur $\mathbb{R}^n$ mais sur $\mathcal{A}^{(k)}$:

$$s^{(k)} = \arg \min_{s \in \mathcal{A}^{(k)}} m^{(k)}(x^{(k)} + s) \tag{6.32}$$

[3] Dans un soucis de simplicité de la présentation et sans perte de généralité, nous supposerons qu'en dehors des composantes fixées, aucune contrainte de borne n'est active au point de départ $x^{(0)}$ bien que l'implémentation pratique de l'algorithme traite sans difficulté cette éventualité.

avec $\|s\| \leq \Delta^{(k)}$. Ce problème reste bel et bien de la forme (6.3) mais avec une dimension inférieure. Il n'y a donc aucun inconvénient à utiliser la méthode de Moré et Sorensen.

Si d'aventure le point $x^{(k)} + s^{(k)}$ est non admissible, celui-ci est projeté sur les contraintes de bornes et le point de test est dés lors défini, composante par composante, par

$$\tilde{x}_i^{(k)} = \begin{cases} \overline{x}_i & \text{si } x_i^{(k)} + s_i^{(k)} \geq \overline{x}_i, \\ \underline{x}_i & \text{si } x_i^{(k)} + s_i^{(k)} \leq \underline{x}_i, \\ x_i^{(k)} + s_i^{(k)} & \text{sinon.} \end{cases} \tag{6.33}$$

Si $\tilde{x}_i^{(k)} = \overline{x}_i$ (resp. $\tilde{x}_i^{(k)} = \underline{x}_i$) et si ça n'était pas le cas à l'itération précédente, la borne supérieure (resp. inférieure) est *activée*, ce qui signifie que l'ensemble

$$\mathcal{F}^{(k+1)} = \mathcal{F}^{(k)} \cup \{i\}. \tag{6.34}$$

Notons que plusieurs variables peuvent éventuellement être activées lors de la même itération.

6.2.5 Critère d'arrêt.

En théorie, l'algorithme procède de la sorte jusqu'à ce que la norme du gradient devienne nulle. En pratique, nous utilisons un seuil numérique défini en fonction de la valeur initiale du gradient, la *réduction relative du gradient*[4]

$$\varepsilon^{(k)} = \frac{\|g^{(k)}\|_{\mathcal{A}^{(k)}}}{\|g^{(0)}\|_{\mathcal{A}^{(0)}}} \tag{6.35}$$

avec

$$\|g\|_{\mathcal{A}^{(k)}} = \sqrt{\sum_{i \notin \mathcal{F}^{(k)}} g_i^2}. \tag{6.36}$$

Soit ε une tolérance fixée *a priori* par l'utilisateur, l'algorithme tente de *désactiver* des contraintes de bornes si

$$\varepsilon^{(k)} \leq \varepsilon \tag{6.37}$$

et si la contrainte $\|s^{(k)}\| = \Delta^{(k)}$ de la région de confiance n'est pas active. L'algorithme ne désactive qu'une seule contrainte à la fois. Pour choisir la contrainte à

[4]Cette façon de procéder est adoptée dans de nombreuses méthodes d'optimisation non-contrainte et présente l'avantage d'effectuer une certaine « mise à échelle » du critère d'arrêt [39, 40].

libérer, il évalue les multiplicateurs de Lagrange relatifs à chacune des variables fixées, c'est-à-dire

$$\mu_i^{(k)} = \begin{cases} -g_i^{(k)} & \text{si } x_i^{(k)} = \underline{x}_i, \\ g_i^{(k)} & \text{si } x_i^{(k)} = \overline{x}_i, \end{cases} \tag{6.38}$$

pour $i \in \mathcal{F}^{(k)} \backslash \mathcal{F}^{(0)}$. Si j est l'indice de la contrainte relative au plus grand multiplicateur de Lagrange

$$\max_{i \in \mathcal{F}^{(k)} \backslash \mathcal{F}^{(0)}} \mu_i^{(k)}, \tag{6.39}$$

et si ce multiplicateur est strictement positif, cette contrainte est désactivée et

$$\mathcal{F}^{(k+1)} = \mathcal{F}^{(k)} \backslash \{j\}. \tag{6.40}$$

Si le plus grand multiplicateur est négatif ou si l'ensemble $\mathcal{F}^{(k)} \backslash \mathcal{F}^{(0)}$ est vide, l'algorithme s'arrête.

6.2.6 Convergence de l'algorithme.

L'objet de la présente section est de vérifier que l'algorithme implémenté répond bel et bien aux hypothèses émises au chapitre 4.

Nous supposerons tout d'abord que les hypothèses portant sur le problème d'optimisation lui-même sont satisfaites : la fonction objectif est deux fois continûment dérivable (hypothèse 4.1), bornée inférieurement (hypothèse 4.2) et son Hessien possède également une borne inférieure au sens d'une certaine norme (hypothèse 4.3).

Les conditions à imposer à l'approximation locale sont, quant à elles, aisément vérifiées : l'approximation locale quadratique est deux fois continûment dérivable (hypothèses 4.4), la fonction objectif et l'approximation locale ainsi que leurs gradients respectifs coïncident bien à l'itération courante (hypothèses 4.5 et 4.6). L'hypothèse 4.7 est satisfaite car, dans l'implémentation, les différentes mises à jour du Hessien de l'approximation locale sont assorties d'un « garde-fou » l'empêchant de prendre des valeurs démesurées.

En vertu du théorème 4.3 dont l'hypothèse est satisfaite en prenant $\kappa_{amm} = 1$, l'hypohèse 4.8 est également satisfaite, *i.e.* la décroissance de l'approximation locale dans la résolution du sous-problème est suffisante puisque le sous-problème est résolu de façon quasi-exacte.

La norme euclidienne est évidemment uniformément équivalente à elle-même, nous pouvons le vérifier facilement en prenant $\kappa_{une} = 1$ (hypothèse 4.9).

L'hypothèse 4.11 de continuité au sens de Lipschitz du Hessien de l'approximation locale est satisfaite étant donné que celui-ci est constant. L'hypothèse 4.12 est automatiquement satisfaite dès lors que le calcul du pas de progression

TAB. 6.2 – Tableau comparatif des différentes versions de Trust.

Version de Trust	Information nécessaire	Convergence vers un pt crit	Découplage souhaitable
SR1 cond.	Gradient	Second ordre	Oui
BFGS cond.	Gradient	Premier ordre	Oui
SR1 incond.	Gradient	Second ordre	Non
BFGS incond.	Gradient	Premier ordre	Non
GN	Jacobienne	Premier ordre	Oui

est effectué en minimisant exactement le sous-problème au sein de la région de confiance. Enfin, la mise à jour du rayon de confiance (6.22) entre bien dans le cadre établi par (4.38) et l'hypothèse 4.13 est donc satisfaite.

En vertu du théorème 4.4, nous pouvons conclure à la convergence vers un point critique du premier ordre de la suite engendrée par l'algorithme.

Si l'hypothèse 4.10 est également satisfaite, nous concluons, en vertu du théorème 4.6, à la convergence globale de la suite engendrée par l'algorithme vers un point critique du second ordre. L'hypothèse 4.10 est généralement remplie pour les méthodes de type SR1, du moins sous les hypothèse du théorème 3.1. Le tableau 6.2 rassemble les différentes propriétés des variantes de Trust comprises dans la présente section et dans la section 6.1.

6.3 Application : identification paramétrique d'un modèle dynamique de type Lotka–Volterra.

Afin d'étudier l'efficacité des méthodes implémentées, nous appliquons celles-ci à l'identification des paramètres d'un système dynamique non-linéaire [101].

6.3.1 Description du modèle.

Penchons-nous sur un des plus vieux exemples de système proie-prédateur étudié : celui de Lotka–Volterra dont les équations sont bien connues en théorie des systèmes non-linéaires [75, 78].

Le modèle de Lotka-Volterra s'écrit sous la forme

$$\begin{aligned} \frac{dX}{dt} &= a_1 X - a_2 XY \\ \frac{dY}{dt} &= a_3 XY - a_4 Y. \end{aligned}$$

TAB. 6.3 – Paramètres de référence pour la génération des mesures par expérience jumelle et estimation initiale des paramètres du modèle qui servent de point de départ à l'optimisation. Rappelons que toutes les simulations sont effectuées avec $\Delta t = 0,05$; $T = 100$; $N_{\max} = 2001$ et $N_{\text{obs}} = 40$.

Paramètre	Valeur initiale	Valeur de référence
$X(0)$	$1,05$	1
$Y(0)$	$1,95$	1
a_1	$0,5$	$0,4$
a_2	$0,3$	$0,2$
a_3	$0,301$	$0,2$
a_4	$0,05$	$0,1$

Dans ce modèle simple, la croissance de la proie X est proportionnelle à sa masse, la prédation est proportionnelle à la masse de la proie et à celle du prédateur Y, cette dernière décroissant par prédation, dégradation (mort) naturelle,... proportionnellement à son importance. Les deux conditions initiales $X(0)$ et $Y(0)$ et les coefficients de proportionnalité a_1, a_2, a_3 et a_4 sont les paramètres du modèle et forment le vecteur des variables de contrôle x.

Le schéma numérique adopté est très simple : il s'agit d'une méthode d'Euler avec évaluation semi-implicite des termes de croissance et de prédation

$$\begin{aligned} \frac{X_{i+1} - X_i}{\Delta t} &= a_1 X_{i+1} - a_2 X_{i+1} Y_i \\ \frac{Y_{i+1} - Y_i}{\Delta t} &= a_3 X_i Y_{i+1} - a_4 Y_{i+1} \end{aligned}$$

avec les conditions initiales $X_0 = X(0)$ et $Y_0 = Y(0)$. La résolution numérique du système est effectuée avec un pas de temps Δt sur un intervalle de temps de longueur T, soit une discrétisation sur $N_{\max}$ points.

Pour réaliser l'identification paramétrique, nous générons des mesures fictives par une expérience jumelle (voir section 5.3). Toutes les expériences jumelles menées dans la suite prennent comme solution de référence les valeurs de paramètres du tableau 6.3. Dans un premier temps, l'identification paramétrique est réalisée avec une estimation initiale arbitraire (voir tableau 6.3). Nous considérons que nous ne disposons que de N_{obs} mesures de la proie X et du prédateur Y à des instants répartis aléatoirement parmi les $N_{\max}$ valeurs échantillonnées numériquement. Les réponses du système sont présentées sur la figure 6.2.

Nous adoptons, pour la quantification de l'erreur du modèle par rapport aux « mesures », l'erreur au sens des moindres carrés (5.9). Si nous définissons la

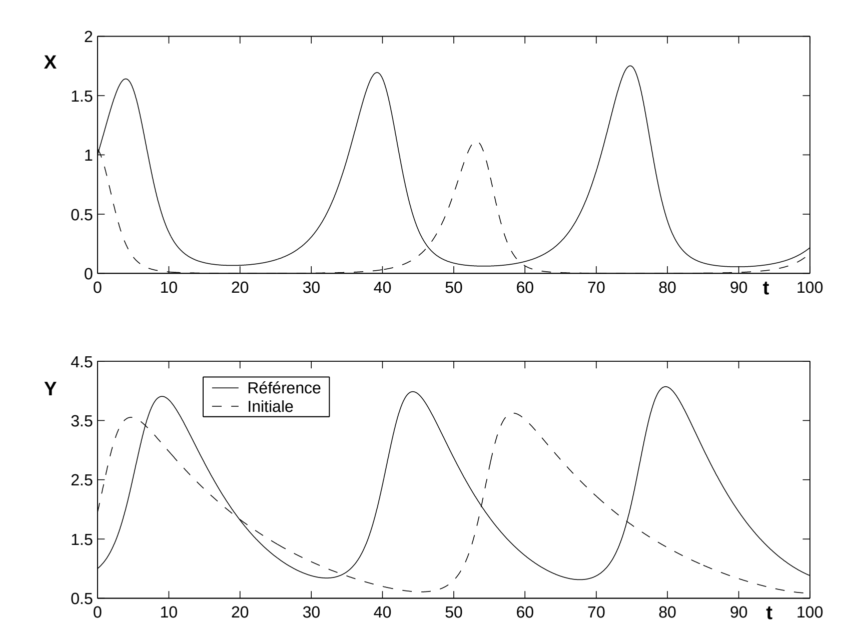


FIG. 6.2 – Réponses temporelles des variables d'état pour la solution de référence et pour l'estimation initiale des variables de contrôle. Le but d'une identification paramétrique est, en termes imagés, de rapprocher les courbes discontinues des courbes continues.

TAB. 6.4 – Coûts CPU des deux méthodes de différentiation normalisés par rapport au coût CPU du modèle. On peut constater que les ordres de grandeur sont bien ceux attendus ($n = 6$).

	Coût CPU	Ordre de grandeur
Différentiation directe	$C_d/C_m = 10,98$	$= O(2n)$
Méthode adjointe	$C_a/C_m = 1,71$	$= O(2)$

fonction d'observation

$$\delta_i = \begin{cases} 1 & \text{si une mesure a été effectuée à l'instant } i, \\ 0 & \text{dans le cas contraire,} \end{cases}$$

nous pouvons écrire la fonction objectif

$$f(x) = \frac{1}{2}\sum_{i=1}^{N_{\max}} \delta_i (X_i - \hat{X}_i)^2 + \frac{1}{2}\sum_{i=1}^{N_{\max}} \delta_i (Y_i - \hat{Y}_i)^2$$

où les variables surmontées d'un chapeau sont les réponses de référence.

La différentiation de ce modèle découle directement des principes énoncés précédemment (voir section 5.4) et ne demande pas de remarques complémentaires. Les deux méthodes de différentiation, directes et adjointes, ont été implémentées. Comme prévu, la première nous donne accès à la matrice jacobienne et au gradient, tandis que la seconde ne fournit que ce dernier. Les coûts CPU des deux méthodes C_d et C_a ont été évalués en terme du coût du modèle direct C_m. On peut constater sur le tableau 6.4 que ceux-ci sont bien de l'ordre de grandeur attendu.

6.3.2 Calibrage du modèle.

Nous allons tester les différentes versions sur ce problème d'identification paramétrique. Pour pouvoir discuter leurs performances, nous utiliserons comme point de comparaison une méthode de type BFGS à mémoire limitée avec une recherche linéaire approchée. Il s'agit de la routine M1QN3 de Gilbert et Lemaréchal [39, 40] qui a déjà montré son efficacité dans le cadre de problèmes d'identification paramétrique [61, 66, 93, 94, 99].

Les résultats obtenus avec M1QN3 sont présentés sur la figure 6.3. On voit que le minimum global est atteint après 79 itérations. Toutefois, les recherches

linéaires portent le nombre d'évaluations de la fonction objectif et de son gradient à 88, auxquelles il convient d'ajouter une simulation initiale du modèle. Puisque seul le gradient est nécessaire à l'utilisation de cette méthode, nous adoptons le modèle adjoint comme méthode de différentiation. Le temps de calcul total de cette routine d'optimisation est donc le suivant[5]

$$C_{\mathsf{M1QN3}} = 89\,(C_m + C_a) = 241,19C_m$$

au vu des rapports du tableau 6.4. Notons que le critère d'arrêt utilisé est identique à celui implémenté par nos soins, la valeur utilisée étant $\varepsilon = 10^{-6}$.

Les différentes variantes de l'algorithme par régions de confiance que nous avons développées ont été appliquées à ce problème type. Les résultats obtenus sont portés sur la figure 6.4. On constate que ces algorithmes se comportent relativement bien, chacune des versions atteignant le minimum global en un nombre raisonnable d'itérations.

On remarque également que les méthodes employant l'actualisation BFGS sont plus rapides que les méthodes SR1. Si les premières ne présentent pas l'assurance de la convergence vers un point critique du second ordre, elles sont néanmoins plus rapides. Ceci est certainement dû au fait que la méthode BFGS donne plus facilement des matrices définies-positives que SR1. Nous attirons l'attention du lecteur sur le fait que, contrairement à son utilisation usuelle, la méthode BFGS ne garantit ici nullement que toutes les approximations quadratiques seront convexes. La propriété bien connue d'hérédité de la définie-positivité n'est en effet valable que dans le cas où les conditions de Wolfe (2.5) et (2.6) sont respectées, ce qui n'est pas nécessairement le cas.

Une autre constatation peut être faite au vu de ces résultats : les méthodes tenant compte d'informations issues des itérations infructueuses — les versions *inconditionnelles* — s'avèrent plus rapides. On peut résumer l'enseignement de cette constatation par la maxime : « on progresse en tirant des leçons de ses erreurs ». Cette approche est analysée en détail au chapitre 7.

Un dernier commentaire sur la méthode Trust-GN : celle-ci paraît être la plus performante puisqu'elle parvient à trouver le minimum global en 40 itérations, soit 8 de moins que sa dauphine Trust-BFGS inconditionnelle. C'est sans compter sur le fait que Trust-GN demande l'évaluation de la matrice jacobienne. Il est donc indispensable d'utiliser la méthode de différentiation directe qui est plus coûteuse que la méthode adjointe utilisée dans les autres tests. Le tableau 5.1 nous enseigne en effet que, pour la méthode de différentiation directe la moins coûteuse[6], il faut

[5] Nous supposons que le temps de simulation du modèle est très grand par rapport à celui de l'optimiseur.

[6] Nous choisissons la méthode de différentiation directe *à mémoire* afin de bénéficier de la propriété qui permet de *découpler* le calcul de la fonction objectif et celui de la matrice jacobienne (voir section 5.4.4).

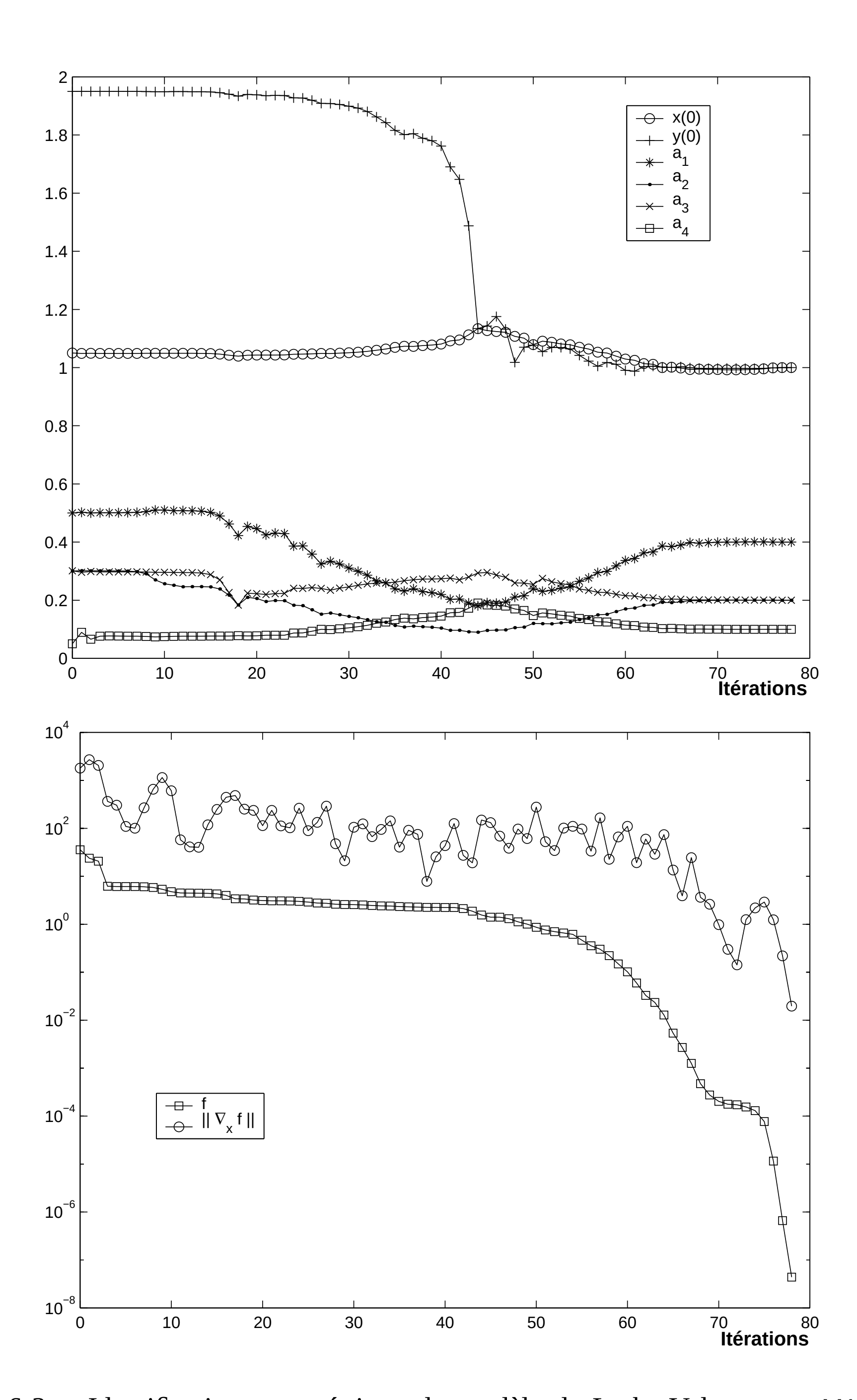


FIG. 6.3 – Identification paramétrique du modèle de Lotka-Volterra par M1QN3. Figure du haut : évolution des paramètres au cours des itérations. Figure du bas : évolution de la fonction objectif et de la norme de son gradient au cours des itérations.

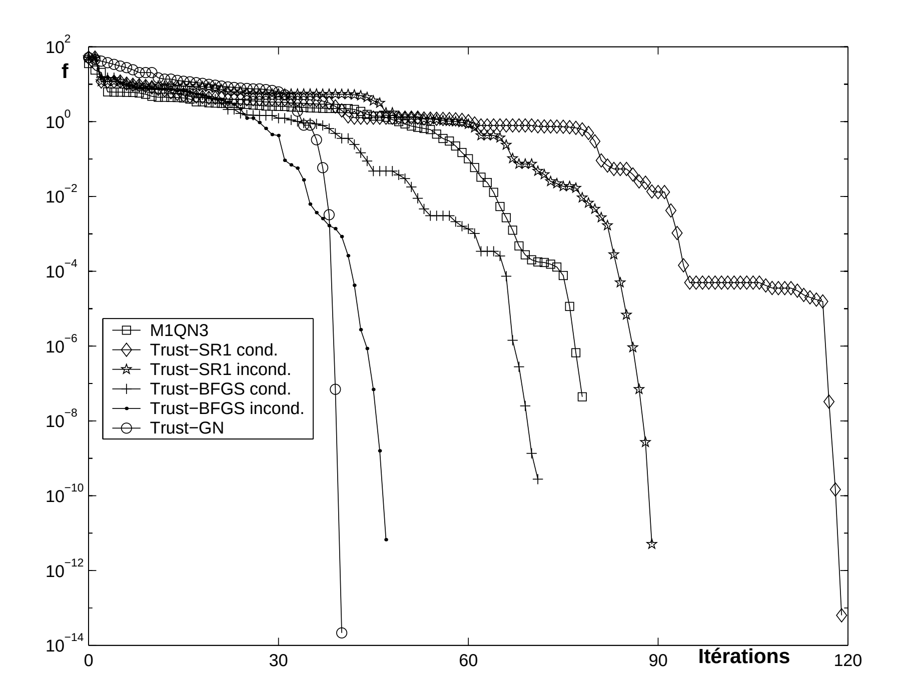


FIG. 6.4 – Fonction objectif $f(x)$ au cours des itérations pour les différentes méthodes d'optimisation.

s'attendre à un coût de l'ordre de $2n$ fois celui du calcul du modèle, ce que nous confirme le tableau 6.4. Ce raisonnement doit se généraliser à toutes les méthodes, puisque les versions *conditionnelles* permettent l'économie d'évaluations du gradient. Le critère de sélection à retenir est donc celui du coût global engendré. Le tableau 6.5 présente celui-ci pour les différentes versions de Trust et pour M1QN3.

On constate que quatre versions de Trust se glissent devant M1QN3 et que Trust-BFGS inconditionnelle est, en fin de compte, la méthode la moins coûteuse. La figure 6.5 illustre son comportement en cours de calcul : on peut y voir l'évolution du rayon de confiance, de la fonction objectif, de la norme de son gradient et des variables.

6.3.3 Analyse statistique.

Afin de fournir une analyse plus détaillée du comportement de Trust pour ce problème, il convient de tester la robustesse des différentes méthodes et d'éviter les hasards malencontreux qui pourraient mener à des conclusions hâtives. Il est évident que, par un agencement fortuit des opérations, une méthode peut paraître plus performante que d'autres en raison du point de départ choisi pour l'optimisation. Pour diminuer l'influence du point de départ sur le calcul, et *de facto* sur

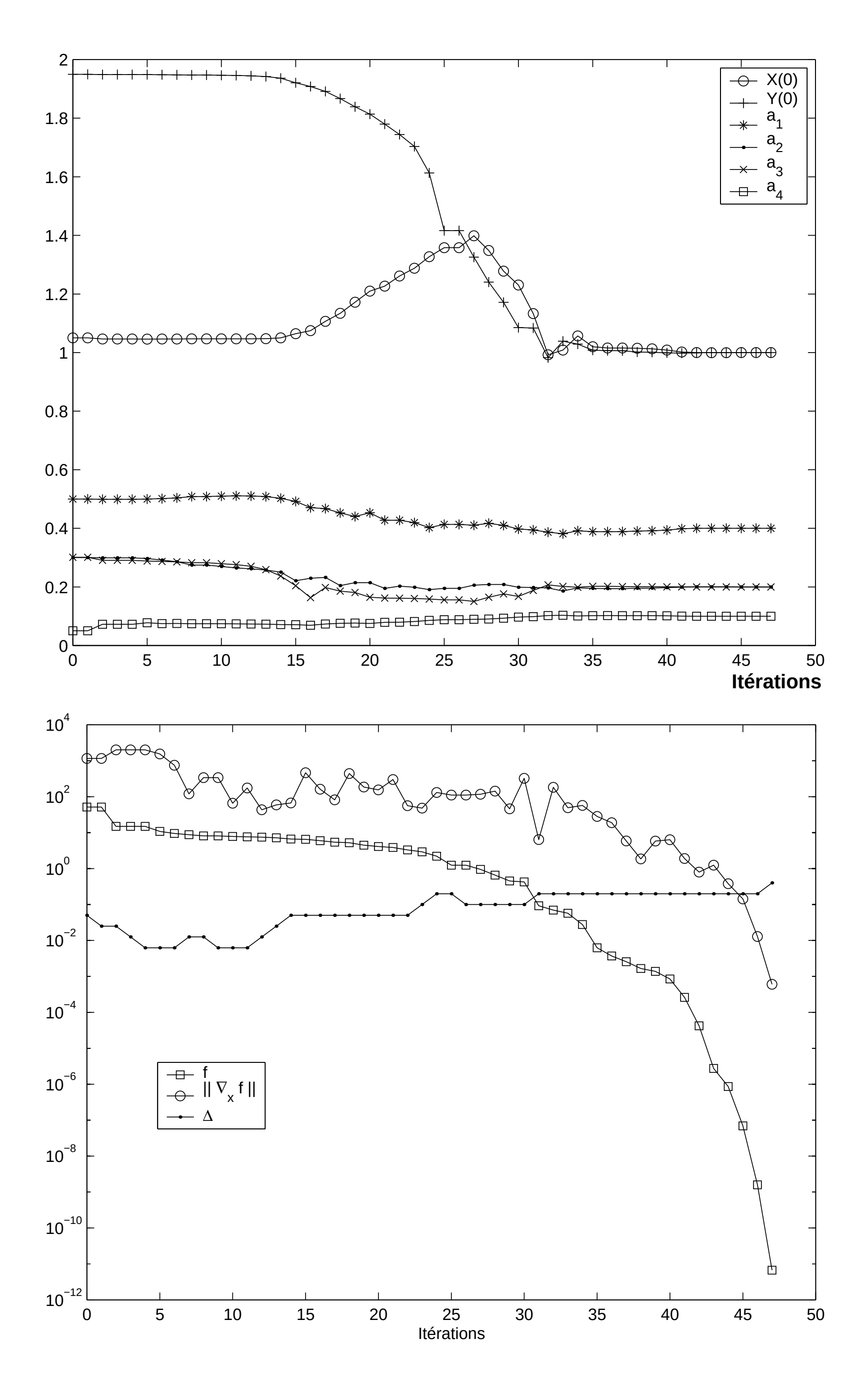


FIG. 6.5 – Identification paramétrique du modèle de Lotka-Volterra par Trust-BFGS inconditionnelle. Figure du haut : évolution des paramètres au cours des itérations. Figure du bas : évolution de la fonction objectif, de la norme de son gradient au cours des itérations et du rayon de confiance.

TAB. 6.5 – Coûts numériques des différentes versions de Trust et M1QN3. Le rayon de confiance initial est identique $\Delta^{(0)} = 0,05$ pour les sept versions. Pour Trust-GN, la troisième colonne indique le nombre d'évaluations de la matrice jacobienne $G(x)$, ce qui explique son coût global important.

Méthode d'optimisation	Évaluations de $f(x)$	Évaluations de $g(x)$ ou $G(x)$	Coût global
Trust-SR1 cond.	120	76	$249,96C_m$
Trust-SR1 incond.	72	72	$195,12C_m$
Trust-BFGS cond.	90	63	$197,73C_m$
Trust-BFGS indond.	48	48	$130,08C_m$
Trust-GN	41	37	$447,26C_m$
M1QN3	89	89	$241,19C_m$

les conclusions, nous avons généré un ensemble de 64 points de départ sur lesquels les informations ont été moyennées. Ces points de départ sont les éléments de l'ensemble de toutes les combinaisons possibles du choix des paramètres x_i, chacun pouvant prendre deux valeurs listées dans le tableau 6.6, soit un total de $2^n = 64$ éléments. Dans les deux valeurs choisies pour chaque paramètre, une est inférieure à la valeur de référence et la seconde supérieure. L'écart par rapport à la valeur de référence est identique de manière à ne « favoriser » aucun point de départ. Ce tableau contient également les valeurs caractéristiques x_i^{car} et les variations caractéristiques δx_i^{car} lorsqu'une mise à échelle est utilisée.

TAB. 6.6 – Ensemble de valeurs ayant servi à la génération de l'ensemble des 64 points de départ. L'ensemble est constitué par les combinaisons possibles de ces paramètres. Les deux dernières colonnes contiennent les valeurs et variations caractéristiques dans les cas où une mise à échelle est utilisée.

Paramètre	Valeur 1	Valeur 2	Val. de réf.	Val. car.	Var. car.
$X(0)$	0,8	1,2	1	1,05	0,3
$Y(0)$	0,8	1,2	1	0,95	0,3
a_1	0,3	0,5	0,4	0,42	0,12
a_2	0,1	0,3	0,2	0,22	0,12
a_3	0,1	0,3	0,2	0,19	0,12
a_4	0,08	0,12	0,1	0,09	0,03

C'est sur base d'une telle analyse que la valeur du paramètre η_3 de la mise à jour du rayon de confiance (6.22) a été choisie. En effet, en comparant celui-ci à

TAB. 6.7 – Tableau comparatif des différentes variantes de Trust. Les trois dernières colonnes sont des moyennes sur toutes les optimisations réussies. À titre de comparaison, M1QN3 a été soumise au même test. Pour Trust-GN, cette colonne indique le nombre d'évaluations de la matrice jacobienne $G(x)$ et non celui du gradient $g(x)$. Ceci explique son coût global étonnamment important en regard du nombre d'itérations relativement faible.

Version de Trust	Notes	Taux de réussite	Évaluations de $f(x)$	Évaluations de $g(x)$ ou $G(x)$	Coût global
SR1 cond.	$\eta_3 = \infty$	50 %	105	62	$211,02C_m$
	$\eta_3 = 1,05$	69 %	102	64	$211,44C_m$
	mise à éch.	81 %	66	43	$139,53C_m$
SR1 incond.	$\eta_3 = \infty$	31 %	65	65	$176,15C_m$
	$\eta_3 = 1,05$	86 %	62	62	$168,02C_m$
	mise à éch.	83 %	41	41	$111,11C_m$
BFGS cond.	$\eta_3 = \infty$	78 %	52	39	$118,69C_m$
	$\eta_3 = 1,05$	78 %	50	40	$118,40C_m$
	mise à éch.	86 %	34	28	$81,88C_m$
BFGS incond.	$\eta_3 = \infty$	80 %	38	38	$102,98C_m$
	$\eta_3 = 1,05$	78 %	38	38	$102,98C_m$
	mise à éch.	88 %	29	29	$78,59C_m$
GN	$\eta_3 = \infty$	67 %	11	10	$120,80C_m$
	$\eta_3 = 1,05$	70 %	12	11	$132,96C_m$
	mise à éch.	72 %	12	11	$132,96C_m$
M1QN3		48 %	56	56	$151,76C_m$

la mise à jour (4.38), on constate que rien ne suggère l'introduction du paramètre η_3 (formellement, on peut considérer que sa valeur est infinie). Pour quelle raison a-t'il dès lors été introduit ? Cette question est traitée en détail au chapitre 7. Nous pouvons néanmoins d'ores et déjà, grâce au tableau 6.7, constater que l'utilisation de $\eta_3 = 1,05$ diminue le coût global et augmente la robustesse[7].

Du tableau 6.7, plusieurs conclusions peuvent être tirées.

– M1QN3, une méthode BFGS avec recherche linéaire, est plus coûteuse que les versions BFGS de l'approche par régions de confiance. La globalisation par régions de confiance apparaît donc, dans ce cas test, plus efficace que l'approche par recherche linéaire.
– Les performances d'une actualisation de type BFGS sont manifestement

[7]Bien que Trust n'ait pas pour vocation d'atteindre le minimum global, nous mesurerons la robustesse en définissant le *taux de réussite* comme le rapport entre le nombre d'optimisations menant à l'optimum global connu et le nombre total d'optimisations tentées (64).

supérieures à celles d'une actualisation de type SR1 qui bénéficie pourtant de propriétés théoriques de convergence plus fortes (voir tableau 6.2). Ce n'est pas surprenant : la mise à jour BFGS est généralement considérée dans la littérature comme la meilleure mise à jour.

- le coût de Trust-GN n'est pas un aussi désastreux que celui que le tableau 6.5 laissait augurer, ce qui justifie *a posteriori* l'utilisation de plusieurs points de départ. Malgré le nombre réduit d'itérations, cette version n'est cependant jamais la meilleure aussitôt que le coût numérique global est pris en compte.
- Les deux versions inconditionnelles qui utilisent l'information issue des itérations infructueuses sont plus efficaces que leurs homologues conditionnelles. L'algorithme est plus efficace quand il apprend de ses erreurs.
- Trust-SR1 inconditionnelle est la méthode la plus robuste avec un taux de réussite de l'ordre de 86% pour un coût de l'ordre de $168,02C_m$. Cette version est donc bien plus robuste que M1QN3 (48%) pour un coût légèrement supérieur ($151,76C_m$ pour M1QN3).
- La méthode la moins coûteuse est Trust-BFGS inconditionnelle avec un ordre de grandeur de $102,98C_m$ pour une robustesse appréciable de 78% qui la place loin devant M1QN3.

Le tableau 6.7 présente également les résultats obtenus avec une version mise à échelle de Trust. Les valeurs et variations caractéristiques utilisées sont celles du tableau 6.6. Comme on pouvait s'y attendre, les versions mises à échelle sont généralement plus robustes et moins coûteuses que leurs homologues non normalisées. Notons que Trust-BFGS inconditionnelle est sans rivale tant du point de vue de la robustesse que du coût global lorsqu'une mise à échelle adéquate est effectuée.

6.4 Conclusion.

Dans le cadre de ce travail, l'algorithme « Trust » a été implémenté dans une routine FORTRAN. Il s'agit d'une méthode utilisant une globalisation par régions de confiance et des approximations locales quadratiques. Différentes versions de type quasi-Newton sont disponibles ainsi qu'une version Gauss-Newton.

L'implémentation est développée en détail : approximations locales, résolution du problème local, critères de convergence, mise à jour du rayon de confiance, calcul du rapport $\rho^{(k)}$, mise à échelle, contraintes de bornes et critère d'arrêt. Le mode d'emploi pratique de la sous-routine écrite en FORTRAN est donné en annexe A.

Enfin, l'algorithme est utilisé pour résoudre un problème d'identification paramétrique d'un système dynamique : le modèle de Lotka–Volterra. Les quelques simulations montrent que la routine Trust est compétitive par rapport à la routine M1QN3 de Gilbert et Lemaréchal [40]. L'approche utilisant une mise à jour

BFGS inconditionnelle de l'approximation de la matrice hessienne s'avère être la plus performante, aussi bien du point de vue de la robustesse que de la vitesse de convergence. Le chapitre suivant montre toutefois l'influence importante de la mise à jour du rayon de confiance sur cette conclusion.

Chapitre 7

La mise à jour du rayon de confiance

Dans les algorithmes d'optimisation par régions de confiance, le rapport $\rho^{(k)}$ défini par (2.29) fournit une mesure de la fidélité de l'approximation locale $m^{(k)}$ à la véritable fonction objectif f dans le voisinage de l'itéré courant. Il est dès lors utilisé pour mettre à jour le rayon de confiance $\Delta^{(k)}$ d'une itération à l'autre. Pour rappel, les règles empiriques habituelles pour cette mise à jour peuvent être résumées par (voir, par exemple, Gould *et al.* [44])

$$\Delta^{(k+1)} = \begin{cases} \alpha_1 \Delta^{(k)} & \text{si } \rho^{(k)} < \eta_1, \\ \Delta^{(k)} & \text{si } \eta_1 \leq \rho^{(k)} < \eta_2, \\ \alpha_2 \Delta^{(k)} & \text{si } \rho^{(k)} \geq \eta_2 \end{cases} \tag{7.1}$$

où α_1, α_2, η_1 and η_2 sont des constantes prédéfinies telles que

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{et} \quad \alpha_1 < 1 < \alpha_2. \tag{7.2}$$

En d'autres mots, le rayon de confiance est réduit après une itération *infructueuse* (*i.e.* $\rho^{(k)} < \eta_1$) et maintenu constant ou augmenté après une itération *réussie* (*i.e.* $\rho^{(k)} \geq \eta_1$).

La stratégie de mise à jour définie par (7.1) est susceptible d'avoir une forte influence sur les performances de l'algorithme. D'une part, les itérés successifs restent proches l'un de l'autre et l'algorithme converge lentement si le rayon de confiance est trop petit. Mais, d'autre part, l'algorithme peut engendrer un grand nombre d'itérations infructueuses si la région de confiance est trop vaste. Ce sujet — critique du point de vue de l'efficacité de l'algorithme — a été relativement peu traité jusqu'à présent. Diverses valeurs pour les paramètres (7.2) ont été utilisées par différents auteurs (*e.g.* Dennis et Mei [23], Gould *et al.* [45]) mais la formule générale (7.1) est rarement mise en cause. Hei [50] généralise (7.1) pour permettre une dépendance continue du rayon de confiance par rapport à $\rho^{(k)}$. Byrd *et al.* [12] suggèrent plusieurs subtilités algorithmiques quand $\rho^{(k)} < 0$, *i.e.* quand le rayon

est trop grand ou quand l'approximation locale est si mauvaise qu'une mesure drastique doit être prise (voir la section 7.3 pour plus de détails).

Dans ce chapitre, nous introduisons une modification de (7.1) qui est d'application lorsque $\rho^{(k)}$ est beaucoup plus grand que l'unité et montrons que, sur un sous-ensemble des problèmes de la collection CUTEr (voir Gould *et al.* [47]), cette modification apporte une amélioration aux performances de l'algorithme. Une généralisation de cette nouvelle stratégie de mise à jour est également développée selon la méthode suivie par Hei [50]. Les travaux exposés dans ce chapitre font l'objet d'une publication (Walmag et Delhez [100]).

7.1 Les itérations trop réussies.

Les itérations de la troisième catégorie définie par (7.1) sont appelées les itération *très réussies* parce qu'elles fournissent des diminutions importantes de la fonction objectif, atteignant au moins les réductions espérées par la minimisation de l'approximation locale $m^{(k)}$. Comme le suggère (7.1), l'approche habituelle dans un tel cas est d'élargir la région de confiance. La raison sous-jacente à cette augmentation de $\Delta^{(k)}$ est intuitive : l'approximation locale semble précise dans une grande région autour de l'itéré courant et l'algorithme devrait dès lors être autorisé à effectuer un pas plus grand si nécessaire.

Cependant, une partie de l'histoire manque. Les itérations dites très réussies usurpent leur nom si la réduction obtenue pour la fonction objectif repose sur une approximation locale imprécise $m^{(k)}$. C'est le cas lorsque $\rho^{(k)}$ est significativement plus grand que l'unité : la réduction de la fonction objectif semble dès lors plutôt fortuite et le risque est grand de pécher par excès de confiance en l'approximation locale $m^{(k)}$. Ceci suggère l'introduction d'un nouvel ensemble d'*itérations trop réussies* caractérisées par $\rho^{(k)} > \eta_3$ où $\eta_3 > 1$ est une constante prédéterminée et le remplacement de (7.1) par la formule (6.22) implémentée dans Trust

$$\Delta^{(k+1)} = \begin{cases} \alpha_1 \Delta^{(k)} & \text{si } \rho^{(k)} < \eta_1, \\ \Delta^{(k)} & \text{si } \eta_1 \leq \rho^{(k)} < \eta_2, \\ \alpha_2 \Delta^{(k)} & \text{si } \eta_2 \leq \rho^{(k)} \leq \eta_3, \\ \alpha_3 \Delta^{(k)} & \text{si } \rho^{(k)} > \eta_3 \end{cases} \tag{7.3}$$

avec

$$0 < \eta_1 \leq \eta_2 < 1 < \eta_3 \tag{7.4}$$

et

$$\alpha_1 < 1 < \alpha_3 < \alpha_2. \tag{7.5}$$

Notons que la stratégie habituelle de mise à jour (7.1) s'avère être un cas particulier de la nouvelle stratégie (7.3) où $\eta_3 = +\infty$.

D'après (7.3), l'accroissement maximum du rayon de confiance se produit lorsque $\rho^{(k)}$ est proche de l'unité, *i.e.* lorsque la fonction $m^{(k)}$ fournit une approximation locale *précise* de la fonction objectif. Dans le cas d'une itération trop réussie, la réduction de la fonction objectif obtenue à l'itération k est significativement plus importante que celle attendue par la minimisation de l'approximation locale $m^{(k)}$. Bien que cette itération permette à l'algorithme de progresser vers un optimum, il n'y a aucune raison de croire que l'itération suivante sera aussi « chanceuse » puisque $m^{(k+1)}$ sera probablement tout aussi imprécis que $m^{(k)}$. Il paraît donc plus prudent d'éviter d'accroître trop rapidement la taille de la région de confiance et de choisir $\alpha_3 < \alpha_2$.

Nous pourrions conclure que la région de confiance doit être rétrécie après une itération trop réussie. Nous choisirons néanmoins $\alpha_3 > 1$ — mais proche de l'unité — pour être conforme à la forme générale de la stratégie de mise à jour (4.38), *i.e.*

$$\Delta^{(k+1)} \in \begin{cases} [\,\gamma_1\Delta^{(k)}, \gamma_2\Delta^{(k)}\,] & \text{if } \rho^{(k)} < \eta_1, \\ [\,\gamma_2\Delta^{(k)}, \Delta^{(k)}\,] & \text{if } \rho^{(k)} \in [\eta_1, \eta_2[, \\ [\,\gamma_3\Delta^{(k)}, \gamma_4\Delta^{(k)}\,] & \text{if } \rho^{(k)} \geq \eta_2. \end{cases} \tag{7.6}$$

déterminée par les constantes réelles η_1, η_2, γ_1, γ_2, γ_3, et γ_4 telles que

$$0 < \eta_1 \leq \eta_2 < 1 \quad \text{et} \quad 0 < \gamma_1 \leq \gamma_2 < 1 < \gamma_3 \leq \gamma_4. \tag{7.7}$$

Bien entendu, la stratégie modifiée (7.3) satisfait aux hypothèses du chapitre 4. Les propriétés générales de convergence globale des algorithmes d'optimisation par régions de confiance sont donc d'application.

7.2 Rayon de confiance auto-adaptatif.

Le concept des itérations trop réussies peut aussi être utilisé dans le cadre des algorithmes au rayon de confiance auto-adaptatif introduits par Hei [50]. L'idée présentée par cet auteur est simplement de faire varier le rayon de confiance $\Delta^{(k+1)}$ plus ou moins continûment avec le rapport $\rho^{(k)}$ suivant

$$\Delta^{(k+1)} = R(\rho^{(k)})\,\Delta^{(k)} \tag{7.8}$$

où R est une fonction telle que les conditions de convergence (7.6) sont satisfaites. Évidemment, la stratégie habituelle (7.1) est un cas particulier de (7.8) correspondant à une fonction étagée R_1 (figure 7.1, en haut à gauche). Hei [50] suggère

d'utiliser des fonctions R non décroissantes telles que

$$R_2(\rho^{(k)}) = \begin{cases} \alpha_1 & \text{si } \rho^{(k)} \leq 0, \\ \alpha_1 + (1-\alpha_1)\left(\frac{\rho^{(k)}}{\eta_2}\right)^2 & \text{si } 0 < \rho^{(k)} < \eta_2, \\ \alpha_3 + (\alpha_2 - \alpha_3)\exp\left\{-\left(\frac{\rho^{(k)}-1}{\eta_2-1}\right)^2\right\} & \text{si } \eta_2 \leq \rho^{(k)} < 1, \\ 2\alpha_2 - \alpha_2 \exp(1-\rho^{(k)}) & \text{si } \rho^{(k)} \geq 1, \end{cases} \tag{7.9}$$

où $\alpha_1 < 1 < \alpha_3 < \alpha_2$ et $\eta_2 < 1$ sont des constantes appropriées (figure 7.1, en bas à gauche). Cette fonction R_2 est qualitativement similaire à la stratégie initiale R_1 puisqu'elle élargit la région de confiance pour les itérations trop réussies.

Pour généraliser la stratégie modifiée (7.3), nous définissons les fonctions Λ comme les fonctions unidimensionnelles $\Lambda(t)$ définies dans $\mathbb{R}$ telles que

1. $\Lambda(t)$ est non décroissante dans $]-\infty, 1]$ et non croissante dans $[1, +\infty[$;
2. $\lim_{t\to-\infty} \Lambda(t) = \alpha_1$;
3. $\lim_{t\to\eta_2^-} \Lambda(t) = 1$;
4. $\Lambda(\eta_2) > \alpha_3 > 1$;
5. $\Lambda(1) = \alpha_2$;
6. $\lim_{t\to+\infty} \Lambda(t) = \alpha_3$

où les constantes α_1, α_2, α_3 satisfont à la condition (7.5) et η_2 est le seuil habituel utilisé dans la définition des itérations très réussies. La stratégie de mise à jour

$$\Delta^{(k+1)} = \Lambda(\rho^{(k)})\, \Delta^{(k)} \tag{7.10}$$

est donc un cas particulier de (7.6) avec $\gamma_1 = \alpha_1$, $\gamma_3 = \alpha_3$, $\gamma_4 = \alpha_2$ et $\gamma_2 = \Lambda(\eta_1)$ de sorte que les propriétés de convergence sont toujours valables. La stratégie modifiée (7.3) peut être décrite par une fonction étagée Λ_1 (figure 7.1, en haut à droite) de la forme (7.10). Comme généralisation de (7.9), nous proposons d'utiliser la fonction Λ définie par

$$\Lambda_2(\rho^{(k)}) = \begin{cases} \alpha_1 & \text{si } \rho^{(k)} \leq 0, \\ \alpha_1 + (1-\alpha_1)\left(\frac{\rho^{(k)}}{\eta_2}\right)^2 & \text{si } 0 < \rho^{(k)} < \eta_2, \\ \alpha_3 + (\alpha_2 - \alpha_3)\exp\left\{-\left(\frac{\rho^{(k)}-1}{\eta_2-1}\right)^2\right\} & \text{si } \rho^{(k)} \geq \eta_2. \end{cases} \tag{7.11}$$

Cette fonction Λ_2 est qualitativement similaire à la stratégie de mise à jour Λ_1 puisqu'elle autorise un élargissement franc de la région de confiance uniquement si $\rho^{(k)}$ est proche de un (figure 7.1, en bas à droite).

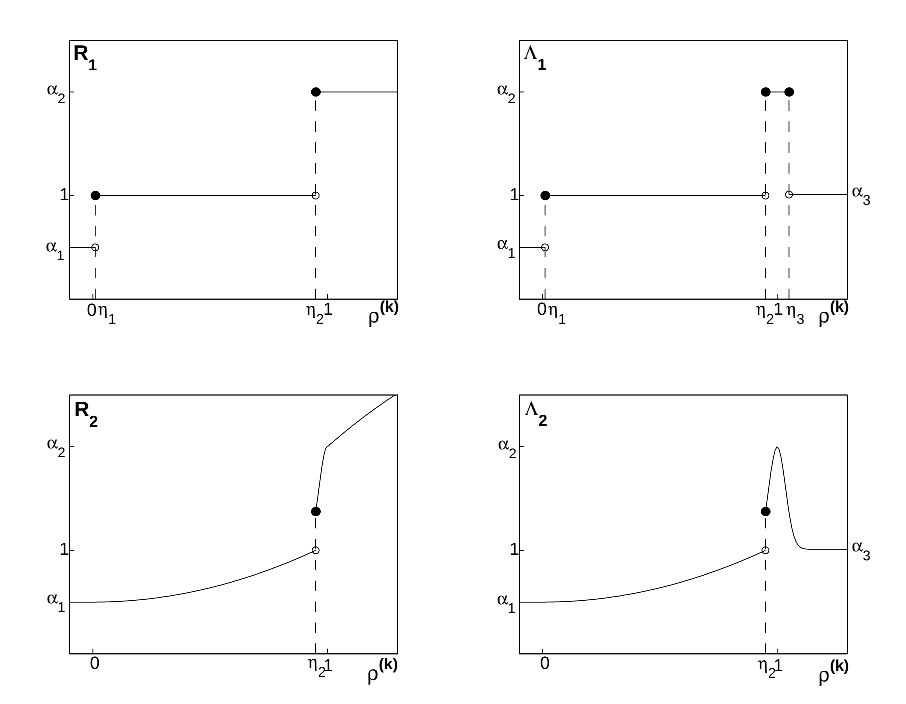


FIG. 7.1 – Rayons de confiance auto-adaptatifs utilisés dans les expériences numériques.

7.3 Raffinements.

Quelques raffinements pour la mise à jour du rayon de confiance ont été introduits dans la littérature spécialisée (voir [20]). Ceux-ci s'avèrent souvent être des astuces empiriques introduites pour augmenter l'efficacité du calcul. Aucune de ces astuces ne semble similaire aux itérations « trop réussies » introduites par Walmag et Delhez [100].

La première règle commode est assez naturelle dès que nous sortons d'un cadre théorique vers l'implémentation numérique : un rayon de confiance maximum $\Delta_{\max}$ est simplement introduit pour éviter les régions de confiance trop grandes.

Un autre raffinement utile est de baser la mise à jour du rayon de confiance sur la longueur du pas $s^{(k)}$. Par exemple, Conn *et al.* [20] ont proposé la règle

$$\Delta^{(k+1)} = \begin{cases} \max(\alpha_2 \|s^{(k)}\|, \Delta^{(k)}) & \text{if } \rho^{(k)} \geq \eta_2, \\ \Delta^{(k)} & \text{if } \rho^{(k)} \in [\eta_1, \eta_2[, \\ \alpha_1 \|s^{(k)}\| & \text{if } \rho^{(k)} \in [0, \eta_1[, \\ \min[\alpha_1 \|s^{(k)}\|, \max(\gamma_1, \gamma_{\text{bad}}^{(k)})\, \Delta^{(k)}] & \text{if } \rho^{(k)} < 0, \end{cases} \tag{7.12}$$

où γ_1 est une constante donnée et

$$\gamma_{\text{bad}}^{(k)} = \frac{(1-\eta_2)\, s^{(k)T} \nabla_x f(x^{(k)})}{(1-\eta_2)[f(x^{(k)}) + s^{(k)T} \nabla_x f(x^{(k)})] + \eta_2 m^{(k)}(\tilde{x}^{(k)}) - f(\tilde{x}^{(k)})}. \tag{7.13}$$

Ces règles peuvent naturellement être appliquées même si les itérations « trop réussies » sont introduites. Cependant, dans un but de clarté de l'exposé, celles-ci n'ont pas été implémentées dans les expériences numériques qui suivent.

7.4 Expériences numériques.

Les idées introduites dans les section 7.1 et 7.2 sont d'abord illustrées sur une variante de la très classique fonction — dite « banane » — de Rosenbrock puis sur quelques problèmes de l'ensemble de problèmes-test CUTEr [5]. Les algorithmes utilisés sont les versions *inconditionnelles* de Trust-BFGS et Trust-SR1 (voir section 6.1 pour plus de détails).

7.4.1 Fonction banane de Rosenbrock.

Un premier test des idées développées dans les sections précédentes peut être effectué sur un problème bien connu : la minimisation de la fonction de Rosenbrock (voir Fletcher [31]) ou, plus exactement, sur une variante logarithmique de

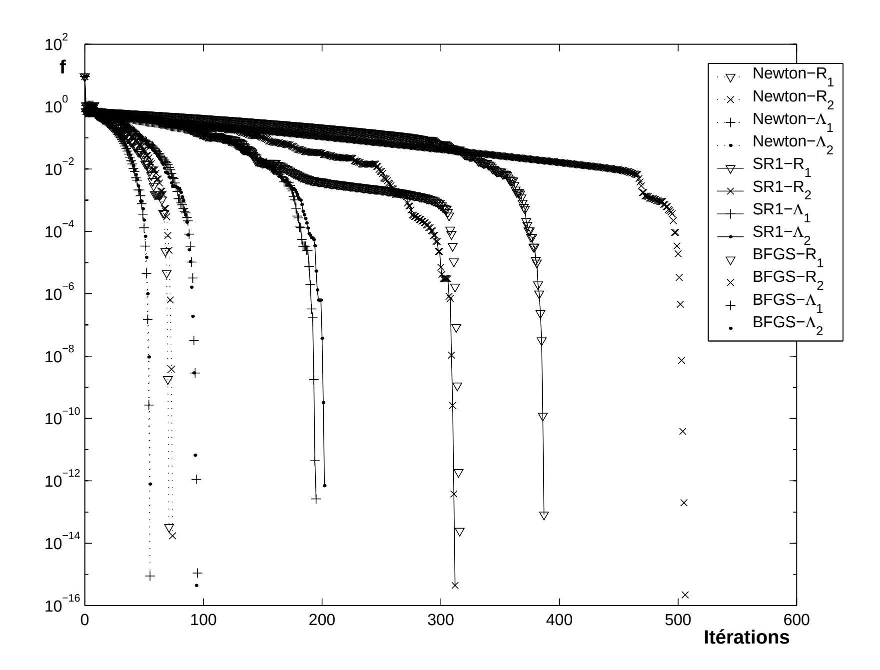


FIG. 7.2 – Évolution de la fonction objectif au cours des itérations. Le point de départ est $(x_1, x_2) = (1, 0)$ et le rayon de confiance initial $\Delta^{(0)} = 1$. Le critère d'arrêt est $\|\nabla_x f(x^{(k)})\| \leq 5 \times 10^{-6}$. Notons que ce sont les versions *inconditionnelles* de Trust-SR1 et Trust-BFGS qui sont utilisées.

celle-ci

$$f(x_1, x_2) = \ln\left[1 + 10000(x_2 - x_1^2)^2 + (1 - x_1)^2\right]. \tag{7.14}$$

Cette fonction présente une vallée courbe et profonde le long de la parabole $x_2 = x_1^2$ et son minimum se situe en $(x_1, x_2) = (1, 1)$. La figure 7.2 nous montre l'évolution de la fonction objectif avec les algorithmes Trust-SR1 et Trust-BFGS associés aux quatre stratégies de mise à jour R_1, R_2, Λ_1 et Λ_2 tandis que les nombres d'itérations requises pour atteindre la convergence sont listés dans le tableau 7.1. Dans un soucis de complétude, nous montrons également le comportement d'une version de Newton utilisant la véritable matrice hessienne.

Aussi bien avec la version Newton qu'avec les versions SR1 et BFGS de l'algorithme, les fonctions Λ se révèlent plus efficaces que les fonctions R. Les modifications algorithmiques suggérées diminuent le nombre d'itérations et, par conséquent, le nombre d'évaluations de la fonction objectif sans aucun calcul supplémentaire. Nous constatons également que la proportion d'itérations réussies est plus grande pour les fonctions Λ.

Le plus petit nombre d'itérations observé avec les fonctions Λ résulte de la combinaison de deux effets. Le premier est une réduction du nombre d'itérations

TAB. 7.1 – Nombre d'itérations pour le problème de Rosenbrock logarithmique. Entre parenthèses : le nombre d'itérations réussies.

	Trust-SR1 incond.		Trust-BFGS incond.		Trust-Newton	
R_1	388	(211)	317	(176)	72	(46)
R_2	312	(153)	506	(260)	74	(46)
Λ_1	195	(152)	96	(87)	55	(48)
Λ_2	203	(164)	94	(86)	56	(49)

induite par la nature conservative de la stratégie de mise à jour ; l'algorithme est ainsi empêché de perdre du temps avec des pas trop grands qui produisent des itérations infructueuses. Le deuxième effet est lié aux mises à jour de la matrice hessienne (inexistante pour Trust-Newton). Les nombreuses itérations infructueuses des versions de l'algorithme utilisant des fonctions R produisent de grands pas $s^{(k)}$ et des mises à jour imprécises de la matrice hessienne. Au contraire, les fonctions Λ donnent des pas $s^{(k)}$ plus courts, et dès lors une mise à jour de type quasi-Newton plus précise, quand l'approximation locale $m^{(k)}(x)$ approche trop grossièrement la fonction objectif $f(x)$.

7.4.2 Performances sur un ensemble de problèmes-tests.

Une comparaison systématique entre les différentes stratégies de mise à jour du rayon de confiance est effectuée pour 70 problèmes de l'ensemble de test CUTEr (voir Bongartz *et al.* [5] et Gould *et al.* [47]). Les problèmes sélectionnés sont tous ceux de petite taille ($n \leq 100$) pour lesquels les dérivées premières sont disponibles ; ils sont répertoriés dans le tableau 7.2. Les résultats sont analysés avec les *profils de performance* proposés par Dolan et Moré [24]. Des profils séparés sont calculés pour les mises à jour SR1 et BFGS.

Pour chacune des deux mises à jour SR1 et BFGS, nous définissons l'*ensemble des problèmes* $\mathcal{P}$ constitué de $n_p (= 70)$ problèmes et l'*ensemble des solveurs* $\mathcal{S}$ constitué des quatre solveurs implémentés avec les différentes stratégies de mise à jour pour le rayon de confiance (R_1, R_2, Λ_1 et Λ_2). Pour chaque problème $p \in \mathcal{P}$ et chaque solveur $s \in \mathcal{S}$, nous évaluons le nombre d'itérations $N_{p,s}$ nécessaires pour résoudre le problème p avec le solveur s. Un *rapport de performance*

$$r_{p,s} = \frac{N_{p,s}}{\min\{N_{p,s} : s \in \mathcal{S}\}}, \tag{7.15}$$

est alors construit en comparant le nombre d'itérations pour résoudre le problème

TAB. 7.2 – Noms et tailles des problèmes de CUTEr sélectionnés. Il s'agit des problèmes de petite taille ($n \leq 100$), différentiables et pour lesquels les dérivées premières sont disponibles.

Nom	*n*	Nom	*n*	Nom	*n*	Nom	*n*
3PK	30	DENSCHNF	2	HYDC20LS	99	PFIT1LS	3
AKIVA	2	DJTL	2	JENSMP	2	PFIT2LS	3
ALLINITU	4	ENGVAL2	3	KOWOSB	4	PFIT3LS	3
BARD	3	EXPFIT	2	LOGHAIRY	2	PFIT4LS	3
BEALE	2	GROWTHLS	3	MARATOSB	2	ROSENBR	2
BIGGS6	6	GULF	3	MEXHAT	2	S308	2
BOX3	3	HAIRY	2	MEYER3	3	SINEVAL	2
BRKMCC	2	HATFLDD	3	OSBORNEA	5	SISSER	2
BROWNBS	2	HATFLDE	3	OSBORNEB	11	SNAIL	2
BROWNDEN	4	HEART6LS	6	PALMER1C	8	STRATEC	10
CLIFF	2	HEART8LS	8	PALMER1D	7	TOINTGOR	50
CUBE	2	HELIX	3	PALMER2C	8	TOINTPSP	50
DECONVU	61	HIELOW	3	PALMER3C	8	TOINTQOR	50
DENSCHNA	2	HIMMELBB	2	PALMER4C	8	VIBRBEAM	8
DENSCHNB	2	HIMMELBF	4	PALMER5C	6	YFITU	3
DENSCHNC	2	HIMMELBG	2	PALMER6C	8	ZANGWIL2	2
DENSCHND	3	HIMMELBH	2	PALMER7C	8		
DENSCHNE	3	HUMPS	2	PALMER8C	8		

p avec le solveur s et les meilleures performances obtenues sur ce problème par un des solveurs de $\mathcal{S}$. Une rapport de performance arbitrairement grand (r_M = 100) est affecté au solveur s si celui-ci s'avère incapable de résoudre un problème donné. Nous définissons le *profil de performance* d'un solveur s comme la distribution cumulée du rapport de performance

$$P_s(\tau) = \frac{1}{n_p} \, |\mathcal{I}_s(\tau)| \tag{7.16}$$

où $\mathcal{I}_s(\tau) = \left\{p \in \mathcal{P} : r_{p,s} \leq \tau\right\}$. La notation $|\mathcal{A}|$ indique le cardinal de l'ensemble $\mathcal{A}$. Avec cette définition, la valeur $P_s(1)$ est la probabilité que le solveur s l'emporte sur les autres solveurs et peut dès lors être utilisée pour comparer les vitesses moyennes des algorithmes. La valeur de la limite

$$P_s^* = \lim_{\tau \to r_M^-} P_s(\tau)$$

est la probabilité pour le solveur s de résoudre un problème et peut dès lors être utilisée pour comparer la robustesse des algorithmes. Ces valeurs sont données au tableau 7.3.

TAB. 7.3 – Vitesse et robustesse des algorithmes pour différentes mises à jour du rayon de confiance.

	Trust-SR1 incond.		Trust-BFGS incond.	
	$P_s(1)$	P_s^*	$P_s(1)$	P_s^*
R_1	$0,30$	$0,74$	$0,43$	$0,83$
R_2	$0,46$	$0,80$	$0,36$	$0,84$
Λ_1	$0,54$	$0,94$	$0,53$	$0,96$
Λ_2	$0,50$	$0,96$	$0,46$	$0,93$

De manière générale, les variantes utilisant les fonctions Λ sont substantiellement plus performantes que celles basées sur les habituelles fonctions R, tant du point de vue de l'efficacité que du point de vue de la robustesse (voir tableaux 7.4 et 7.5). Cette plus grande efficacité des versions Λ est clairement démontrée par les tracés des profils de performance complets (figure 7.3) : les courbes correspondant aux fonctions Λ se situent sous les versions utilisant les fonctions R pour toutes les valeurs de τ.

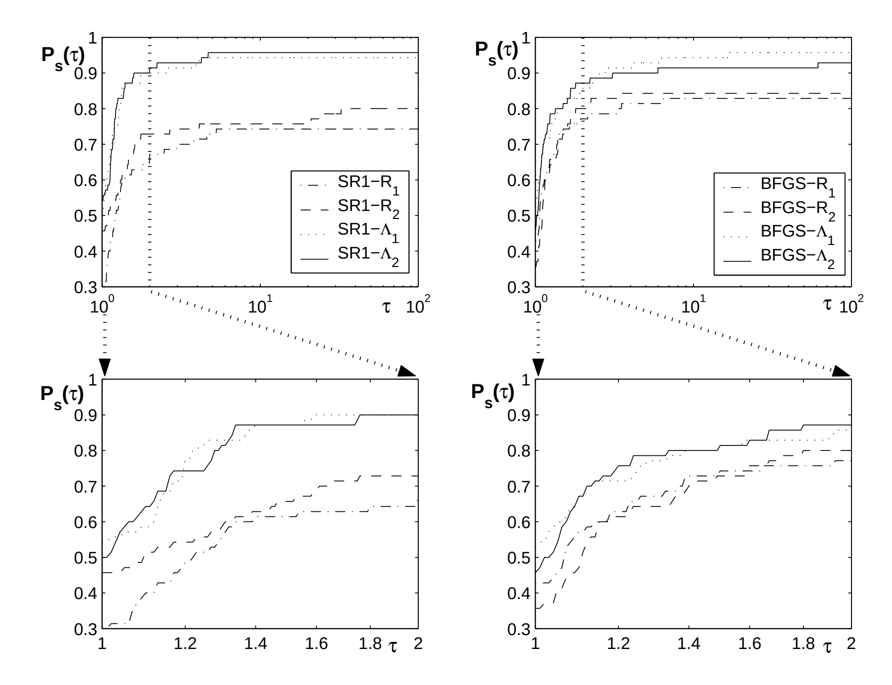


FIG. 7.3 – Profils de performance des différentes versions de l'algorithme pour 70 problèmes de la collection CUTEr. Les expériences numériques utilisent les valeurs du tableau 6.1, un rayon de confiance initial $\Delta^{(0)} = 1$ et un critère d'arrêt $\|\nabla_x f(x^{(k)})\| / \|\nabla_x f(x^{(0)})\| \leq 10^{-6}$. Les deux figures du bas sont des zooms des deux figures du haut.

TAB. 7.4 – Résultats détaillés pour Trust-SR1 avec mise à jour quasi-Newton inconditionnelle. Nombre d'itérations pour chacun des problèmes sélectionnés. Entre parenthèses : nombre d'itérations réussies. Le symbole « – » signifie que le point de test $\tilde{x}^{(k)}$ produit un dépassement de valeur pour la fonction objectif ou pour son gradient. Le symbole « ∗∗ » signifie que le nombre d'itérations dépasse 10.000. Une étoile en exposant signifie que la convergence se fait vers un autre minimum local avec une valeur plus grande de la fonction objectif.

Nom	R_1		R_2		Λ_1		Λ_2	
3PK	66	(50)	71	(49)	68	(50)	67	(46)
AKIVA	20	(13)	19	(12)	15	(11)	15	(11)
ALLINITU	10	(9)	14	(11)	10	(9)	12	(10)
BARD	19	(14)	13	(13)	16	(14)	12	(12)
BEALE	26	(19)	17	(15)	17	(15)	17	(16)
BIGGS6	–		57	(38)	46	(38)	44	(39)
BOX3	9	(9)	18	(11)	9	(9)	14	(12)
BRKMCC	5	(4)	5	(4)	5	(4)	5	(4)
BROWNBS	50	(40)	50	(41)	43	(37)	49	(44)
BROWNDEN	24	(21)	25	(21)	19	(18)	23	(21)
CLIFF	1	(1)	1	(1)	1	(1)	1	(1)
CUBE	∗∗		∗∗		58	(42)	58	(51)
DECONVU	62	(42)	68	(50)	63	(43)	71	(58)
DENSCHNA	9	(9)	9	(9)	9	(9)	9	(9)
DENSCHNB	11	(10)	10	(10)	10	(10)	10	(10)
DENSCHNC	14	(14)	13	(12)	14	(14)	13	(12)
DENSCHND	23	(23)	22	(22)	28	(26)	23	(23)
DENSCHNE	–		∗∗		22	(21)	30	(29)
DENSCHNF	9	(9)	11	(10)	9	(9)	10	(9)
DJTL	5369	(2695)	∗∗		256	(171)	∗∗	
ENGVAL2	53	(40)	∗∗		33	(28)	32	(27)
EXPFIT	∗∗		∗∗		21	(16)	16	(14)
GROWTHLS	29*	(19*)	–		51	(37)	46	(39)
GULF	63	(41)	–		43	(37)	53	(45)
HAIRY	44	(30)	52	(34)	56	(50)	206	(194)
HATFLDD	24	(20)	∗∗		27	(22)	28	(26)
HATFLDE	18	(15)	32	(23)	13	(10)	26	(21)
HEART6LS	∗∗		∗∗		∗∗		5070	(4059)
HEART8LS	∗∗		∗∗		∗∗		1492	(1196)
HELIX	35	(26)	80	(49)	34	(26)	27	(23)
HIELOW	15	(11)	16	(12)	20	(16)	17	(13)
HIMMELBB	3	(3)	3	(3)	3	(3)	3	(3)
HIMMELBF	20	(18)	36	(26)	35	(27)	28	(26)

suite à la page suivante . . .

TAB. 7.4 – (suite du tableau)

... suite de la page précédente

Nom	R_1		R_2		Λ_1		Λ_2	
HIMMELBG	11	(8)	12	(9)	10	(7)	9	(7)
HIMMELBH	7	(7)	7	(7)	7	(7)	7	(7)
HUMPS	221	(138)	178	(118)	138	(108)	307	(266)
HYDC20LS	228	(166)	160	(131)	209	(165)	158	(136)
JENSMP	$**$		$**$		38	(31)	33	(29)
KOWOSB	83	(49)	$**$		31	(26)	31	(22)
LOGHAIRY	276	(187)	1267	(912)	$**$		$**$	
MARATOSB	9	(7)	8	(6)	9	(7)	8	(6)
MEXHAT	19	(17)	19	(17)	19	(17)	19	(17)
MEYER3	33^*	(21^*)	35^*	(24^*)	38	(32)	40	(32)
OSBORNEA	–		–		–		–	
OSBORNEB	62	(41)	81	(57)	80	(61)	78	(61)
PALMER1C	7	(7)	7	(7)	7	(7)	7	(7)
PALMER1D	8	(8)	9	(9)	8	(8)	9	(9)
PALMER2C	7	(7)	7	(7)	7	(7)	7	(7)
PALMER3C	7	(7)	7	(7)	7	(7)	7	(7)
PALMER4C	7	(7)	7	(7)	7	(7)	7	(7)
PALMER5C	7	(7)	7	(7)	7	(7)	7	(7)
PALMER6C	8	(8)	9	(9)	8	(8)	9	(9)
PALMER7C	5	(5)	6	(6)	5	(5)	6	(6)
PALMER8C	6	(6)	7	(7)	7	(7)	7	(7)
PFIT1LS	–		–		651	(516)	255	(209)
PFIT2LS	–		–		198	(155)	48	(40)
PFIT3LS	–		–		515	(408)	502	(406)
PFIT4LS	–		–		816	(622)	755	(590)
ROSENBR	1399	(706)	$**$		43	(33)	59	(49)
S308	11	(11)	10	(10)	11	(11)	12	(12)
SINEVAL	3613	(1814)	535	(246)	141	(108)	159	(129)
SISSER	9	(9)	9	(9)	9	(9)	9	(9)
SNAIL	185	(98)	45	(26)	177	(149)	191	(173)
STRATEC	$**$		91	(59)	87	(63)	75	(58)
TOINTGOR	45	(32)	48	(40)	46	(35)	49	(41)
TOINTPSP	86	(49)	75	(46)	62	(44)	49	(37)
TOINTQOR	25	(24)	25	(22)	25	(24)	25	(22)
VIBRBEAM	72	(33)	110	(68)	58	(29)	55	(31)
YFITU	$**$		860	(433)	163	(122)	194	(150)
ZANGWIL2	2	(2)	2	(2)	2	(2)	2	(2)

fin du tableau

TAB. 7.5 – Résultats détaillés pour Trust-BFGS avec mise à jour quasi-Newton inconditionnelle. Nombre d'itérations pour chacun des problèmes sélectionnés. Entre parenthèses : nombre d'itérations réussies. Le symbole « – » signifie que le point de test $\tilde{x}^{(k)}$ produit un dépassement de valeur pour la fonction objectif ou pour son gradient. Le symbole « ∗∗ » signifie que le nombre d'itérations dépasse 10.000. Une étoile en exposant signifie que la convergence se fait vers un autre minimum local avec une valeur plus grande de la fonction objectif.

Nom	R_1		R_2		Λ_1		Λ_2	
3PK	154	(117)	116	(96)	183	(157)	117	(99)
AKIVA	18	(11)	18	(11)	16	(12)	13	(9)
ALLINITU	11	(10)	10	(9)	11	(10)	10	(9)
BARD	16	(15)	16	(16)	16	(15)	16	(16)
BEALE	13	(13)	16	(16)	13	(13)	13	(13)
BIGGS6	41	(38)	42	(38)	38	(36)	41	(40)
BOX3	9	(9)	15	(15)	9	(9)	15	(15)
BRKMCC	6	(5)	6	(5)	6	(5)	6	(5)
BROWNBS	35	(34)	52	(45)	35	(34)	42	(38)
BROWNDEN	24	(19)	24	(19)	23	(19)	24	(20)
CLIFF	1	(1)	1	(1)	1	(1)	1	(1)
CUBE	∗∗		67	(59)	49	(44)	39	(36)
DECONVU	100	(81)	105	(103)	101	(80)	102	(99)
DENSCHNA	9	(9)	9	(9)	9	(9)	9	(9)
DENSCHNB	9	(9)	9	(9)	9	(9)	9	(9)
DENSCHNC	13	(13)	14	(14)	13	(13)	14	(14)
DENSCHND	15	(14)	21	(20)	20	(19)	25	(24)
DENSCHNE	–		–		34	(33)	37	(35)
DENSCHNF	8	(8)	8	(8)	8	(8)	8	(8)
DJTL	∗∗		∗∗		∗∗		∗∗	
ENGVAL2	33	(29)	31	(28)	29	(26)	27	(25)
EXPFIT	17	(16)	16	(14)	17	(15)	15	(14)
GROWTHLS	–		–		196	(172)	48	(46)
GULF	∗∗		23	(17)	51	(40)	51	(44)
HAIRY	52	(34)	70	(45)	99	(87)	160	(150)
HATFLDD	22	(21)	23	(23)	22	(21)	25	(25)
HATFLDE	29	(27)	21	(21)	29	(27)	22	(22)
HEART6LS	∗∗		∗∗		1634	(1457)	∗∗	
HEART8LS	∗∗		∗∗		876	(801)	346	(324)
HELIX	21	(18)	24	(23)	22	(20)	26	(24)
HIELOW	20	(15)	17	(13)	19	(14)	18	(14)
HIMMELBB	3	(3)	3	(3)	3	(3)	3	(3)
HIMMELBF	34	(33)	36	(35)	35	(34)	32	(31)

suite à la page suivante . . .

TAB. 7.5 – (suite du tableau)

... suite de la page précédente

Nom	R_1		R_2		Λ_1		Λ_2	
HIMMELBG	11	(9)	11	(9)	9	(7)	9	(7)
HIMMELBH	8	(7)	8	(7)	8	(7)	8	(7)
HUMPS	428	(262)	258	(157)	122	(90)	7496	(7441)
HYDC20LS	347	(286)	369	(341)	347	(286)	369	(341)
JENSMP	**		–		45	(39)	36	(32)
KOWOSB	30	(28)	30	(28)	30	(28)	33	(32)
LOGHAIRY	449	(300)	491	(331)	**		**	
MARATOSB	27	(20)	19	(12)	27	(20)	14	(8)
MEXHAT	14	(12)	14	(12)	14	(12)	14	(12)
MEYER3	83*	(66*)	66*	(57*)	415	(387)	394	(386)
OSBORNEA	–		–		–		–	
OSBORNEB	62	(53)	63	(58)	61	(54)	57	(53)
PALMER1C	20	(17)	32	(29)	20	(17)	32	(29)
PALMER1D	20	(19)	21	(19)	19	(18)	22	(20)
PALMER2C	16	(15)	15	(14)	16	(15)	15	(14)
PALMER3C	16	(15)	27	(26)	15	(14)	27	(26)
PALMER4C	12	(11)	19	(18)	12	(11)	18	(17)
PALMER5C	18	(13)	20	(17)	18	(13)	20	(17)
PALMER6C	18	(17)	12	(11)	18	(17)	12	(11)
PALMER7C	15	(14)	11	(10)	14	(13)	11	(10)
PALMER8C	17	(16)	19	(18)	16	(15)	19	(18)
PFIT1LS	–		–		449	(402)	27	(23)
PFIT2LS	154	(124)	–		357	(326)	**	
PFIT3LS	–		–		921	(846)	332	(315)
PFIT4LS	–		–		470	(432)	522	(498)
ROSENBR	118	(68)	40	(35)	35	(31)	35	(31)
S308	13	(13)	13	(13)	13	(13)	16	(14)
SINEVAL	187	(128)	195	(124)	89	(76)	87	(80)
SISSER	9	(9)	9	(9)	9	(9)	9	(9)
SNAIL	632	(328)	343	(186)	98	(92)	103	(98)
STRATEC	72	(63)	80	(71)	72	(63)	79	(71)
TOINTGOR	75	(61)	76	(73)	75	(61)	76	(74)
TOINTPSP	62	(44)	60	(50)	62	(44)	62	(51)
TOINTQOR	47	(29)	42	(40)	47	(29)	42	(40)
VIBRBEAM	112	(81)	68	(50)	70	(52)	91	(71)
YFITU	89	(75)	107	(82)	76	(67)	80	(72)
ZANGWIL2	2	(2)	2	(2)	2	(2)	2	(2)

fin du tableau

7.5 Interaction avec la stratégie de mise à jour de type quasi-Newton.

Comme mentionné plus haut dans l'analyse du problème de Rosenbrock (section 7.4.1), il y a une nette interaction entre la stratégie de mise à jour de la matrice hessienne et la stratégie de mise à jour du rayon de la région de confiance. Les résultats obtenus dans les expériences numériques de la section précédente utilisent une mise à jour de type quasi-Newton inconditionnelle, *i.e.* l'approximation de la matrice hessienne est mise à jour à chaque itération, que celle-ci soit réussie ou non. Une telle stratégie de mise à jour trouve sa justification dans l'espoir d'améliorer la convergence en utilisant toute l'information disponible aux points de test successifs. Cependant, cette approche dégrade les performances de l'algorithme dans le cas de grands pas de progression qui causent une « pollution » de la matrice hessienne avec des mises à jour de piètre qualité qui sont difficiles à compenser. C'est particulièrement vrai lorsque le rayon de confiance initial est bien trop grand pour le problème envisagé. Grâce à leur nature conservative, les fonctions Λ génèrent moins facilement de tels pas de progression trop grands et rendent dès lors moins probable la pollution de la matrice hessienne.

7.5.1 La règle empirique de Byrd *et al.*

Des solutions empiriques ont été proposées pour empêcher ces mises à jour quasi-Newton imprécises dues à de trop grands pas de progression. Byrd *et al.* [12] suggèrent de ne pas effectuer la mise à jour pour les itérations où la variation de f est trop grande, *i.e.* quand

$$f(x^{(k)}+s^{(k)})-f(x^{(k)})>0,5\left[f(x^{(0)})-f(x^{(k)})\right]. \tag{7.17}$$

Implémenter cette condition (7.17) avec les fonctions R n'apporte aucune amélioration dans le problème de Rosenbrock : la décroissance de la fonction objectif à la première itération est tellement importante que (7.17) n'est jamais activée. Cette règle empirique est très sensible à la valeur initiale de la fonction objectif $f(x^{(0)})$. Telle qu'utilisée par Byrd *et al.* [12], (7.17) transforme la première itération en une recherche linéaire (arrière) le long de la direction de plus grande pente. En effet, démarrant avec une matrice identité comme estimation initiale de la matrice hessienne, les points de tests générés subséquemment restent le long de la direction de plus grande pente tant qu'aucune mise à jour de la matrice $H^{(k)}$ n'est effectuée.

Si nous utilisons $f(x^{(1)})$ en lieu et place de $f(x^{(0)})$ dans (7.17) dans le cadre du problème de Rosenbrock, la mise à jour de la matrice hessienne n'est pas effectuée de la deuxième itération jusqu'à la huitième et la convergence est obtenue

après 345 itérations avec l'algorithme Trust-BFGS-R_2 (contre 506 sans utiliser la règle (7.17) *cf.* tableau 7.1). Bien que la vitesse de convergence soit considérablement améliorée, les effets positifs disparaissent après la huitième itération : les diminutions de la fonction objectif $f(x^{(1)}) - f(x^{(k)})$ sont trop grandes pour encore activer (7.17) avant l'arrêt de l'algorithme.

L'introduction des fonctions Λ a essentiellement le même objectif que la règle empirique (7.17) mais y parvient de façon plus efficace. Pour le problème de Rosenbrock, la condition (7.17) n'est jamais activée lorsque des fonctions Λ sont utilisées pour mettre à jour le rayon de confiance. En conséquence, la convergence est atteinte en 94 itérations avec la fonction Λ_2. De plus, cette nouvelle stratégie de mise à jour du rayon de confiance ne souffre pas de la même dépendance critique par rapport au point de départ $x^{(0)}$ que la règle (7.17).

7.5.2 Mise à jour conditionnelle de la matrice hessienne.

Une autre approche largement utilisée pour éviter les mises à jour médiocres de l'approximation de la matrice hessienne est, tout simplement, de n'effectuer la mise à jour que pour les itérations réussies (voir la discussion dans Byrd *et al.* [12]). Cette approche, que nous nommerons *mise à jour conditionnelle*, est aisément implémentée. Comme le montrent les résultats de son application aux 70 problèmes-tests utilisés précédemment (tableau 7.6), cette stratégie de mise à jour conditionnelle est à la fois robuste et efficace : utilisée avec les fonctions R, elle induit une diminution drastique du nombre d'itérations par rapport à la stratégie inconditionnelle correspondante. De ce point de vue, elle offre une alternative aux fonctions Λ.

Nous pouvons maintenant combiner les différentes stratégies de mise à jour de l'approximation de la matrice hessienne et du rayon de confiance. Les résultats détaillés de ces combinaisons utilisant l'approche *conditionnelle* de la mise à jour quasi-Newton sont listés dans le tableau 7.6 et peuvent être comparés aux résultats des tableaux 7.4 et 7.5. Pour comparer les différentes combinaisons, de nouveaux profils de performances sont calculés séparément pour les stratégies de mise à jour SR1 et BFGS. Pour simplifier l'analyse, seules les fonctions étagées R_1 et Λ_1 sont considérées, en combinaison avec les approches conditionnelle et inconditionnelle de la mise à jour quasi-Newton.

Les profils de performance (figure 7.4) confirment les meilleurs résultats des fonctions Λ par rapport aux fonctions R dans le cas inconditionnel. Avec les versions R_1, l'approche conditionnelle se comporte également mieux que la stratégie inconditionnelle, la première citée se révélant plus robuste que la seconde. Notons cependant que la vitesse moyenne de convergence de l'approche inconditionnelle est significativement plus importante dans le cas d'une mise à jour de type BFGS. De ces quatre algorithmes, la combinaison de la fonction Λ_1 pour la mise à jour

du rayon de la région de confiance et de la mise à jour quasi-Newton inconditionnelle peut être recommandée. Bien que légèrement moins robuste que les deux variantes conditionnelles, elle est significativement plus rapide que les trois autres algorithmes.

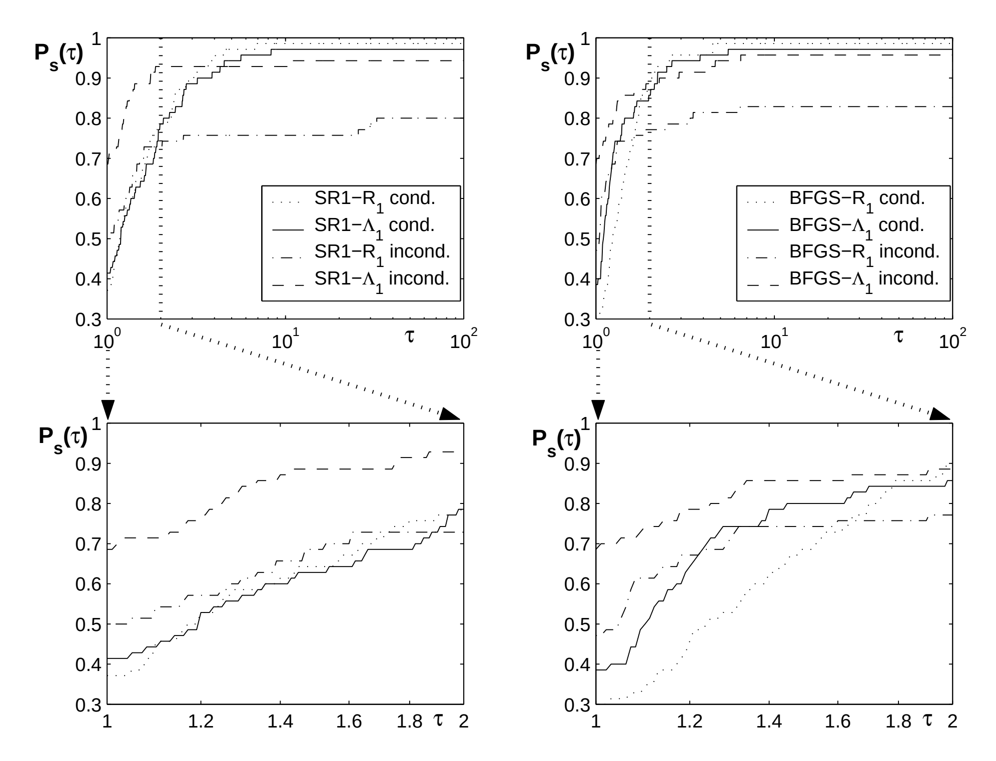


FIG. 7.4 – Profils de performance des différentes versions de l'algorithme pour 70 problèmes de la collection CUTEr. Les expériences numériques utilisent les valeurs du tableau 6.1, un rayon de confiance initial $\Delta^{(0)} = 1$ et un critère d'arrêt $\|\nabla_x f(x^{(k)})\| / \|\nabla_x f(x^{(0)})\| \leq 10^{-6}$. Les deux figures du bas sont des zooms des deux figures du haut.

TAB. 7.6 – Résultats détaillés pour l'approche conditionnelle de mise à jour du Hessien (SR1 et BFGS) et les fonctions R_1 et Λ_1 pour la mise à jour du rayon de confiance. Nombre d'itérations pour chacun des problèmes sélectionnés. Entre parenthèses : nombre d'itérations réussies. Le symbole « − » signifie que le point de test $\tilde{x}^{(k)}$ produit un dépassement de valeur pour la fonction objectif ou pour son gradient. Le symbole « ∗∗ » signifie que le nombre d'itérations dépasse 10.000. Une étoile en exposant signifie que la convergence se fait vers un autre minimum local avec une valeur plus grande de la fonction objectif.

Nom	SR1-R_1		SR1-Λ_1		BFGS-R_1		BFGS-Λ_1	
3PK	57	(35)	57	(35)	166	(143)	147	(123)
AKIVA	18	(10)	18	(11)	18	(10)	18	(11)
ALLINITU	12	(10)	12	(10)	14	(11)	14	(11)
BARD	14	(13)	13	(12)	17	(16)	17	(16)
BEALE	23	(15)	23	(16)	13	(13)	13	(13)
BIGGS6	57	(31)	55	(34)	50	(36)	45	(40)
BOX3	9	(9)	9	(9)	9	(9)	9	(9)
BRKMCC	5	(4)	5	(4)	6	(5)	6	(5)
BROWNBS	159	(96)	114	(72)	69	(49)	69	(48)
BROWNDEN	22	(16)	21	(18)	40	(25)	28	(21)
CLIFF	1	(1)	1	(1)	1	(1)	1	(1)
CUBE	90	(53)	96	(60)	64	(43)	45	(37)
DECONVU	171	(103)	128	(83)	124	(101)	124	(101)
DENSCHNA	9	(9)	9	(9)	9	(9)	9	(9)
DENSCHNB	11	(10)	10	(10)	9	(9)	9	(9)
DENSCHNC	14	(14)	14	(14)	13	(13)	13	(13)
DENSCHND	23	(23)	24	(22)	21	(18)	21	(20)
DENSCHNE	28	(19)	27	(22)	−		34	(29)
DENSCHNF	9	(9)	9	(9)	8	(8)	8	(8)
DJTL	189	(104)	180	(98)	188	(100)	188	(100)
ENGVAL2	79	(42)	60	(38)	29	(26)	31	(27)
EXPFIT	15	(13)	15	(13)	13	(11)	13	(11)
GROWTHLS	55	(32)	70	(44)	42	(26)	161	(141)
GULF	94	(53)	242	(129)	45	(33)	46	(41)
HAIRY	79	(57)	141	(127)	102	(70)	115	(101)
HATFLDD	97	(47)	93	(47)	24	(23)	24	(23)
HATFLDE	42	(24)	35	(23)	22	(21)	22	(21)
HEART6LS	9552	(5247)	∗∗		2518	(1589)	∗∗	
HEART8LS	593	(365)	414	(276)	685	(456)	386	(323)
HELIX	27	(24)	34	(27)	34	(22)	24	(21)
HIELOW	43	(15)	23	(14)	21	(16)	21	(16)
HIMMELBB	3	(3)	3	(3)	3	(3)	3	(3)

suite à la page suivante . . .

TAB. 7.6 – (suite du tableau)

... suite de la page précédente

Nom	SR1-R_1		SR1-Λ_1		BFGS-R_1		BFGS-Λ_1	
HIMMELBF	22	(20)	37	(29)	32	(31)	34	(33)
HIMMELBG	14	(7)	13	(7)	12	(7)	11	(7)
HIMMELBH	7	(7)	7	(7)	10	(8)	10	(8)
HUMPS	274	(170)	227	(173)	143	(90)	247	(185)
HYDC20LS	112	(91)	112	(91)	296	(237)	296	(237)
JENSMP	60	(28)	43	(31)	46	(32)	50	(39)
KOWOSB	36	(22)	60	(36)	51	(30)	38	(33)
LOGHAIRY	**		**		2027	(1360)	**	
MARATOSB	13	(6)	12	(6)	122	(78)	149	(125)
MEXHAT	31	(18)	27	(18)	23	(16)	23	(16)
MEYER3	222	(108)	274	(146)	124	(75)	64	(49)
OSBORNEA	36	(17)	33	(18)	88	(55)	73	(57)
OSBORNEB	142	(86)	100	(68)	73	(58)	68	(58)
PALMER1C	7	(7)	7	(7)	34	(23)	34	(23)
PALMER1D	8	(8)	8	(8)	42	(29)	42	(29)
PALMER2C	7	(7)	7	(7)	34	(22)	34	(22)
PALMER3C	7	(7)	7	(7)	39	(26)	40	(27)
PALMER4C	7	(7)	7	(7)	30	(20)	30	(20)
PALMER5C	7	(7)	7	(7)	26	(19)	26	(19)
PALMER6C	8	(8)	8	(8)	14	(13)	14	(13)
PALMER7C	5	(5)	5	(5)	19	(17)	16	(15)
PALMER8C	6	(6)	7	(7)	21	(19)	22	(20)
PFIT1LS	77	(40)	62	(35)	449	(299)	274	(246)
PFIT2LS	900	(506)	897	(532)	108	(69)	81	(66)
PFIT3LS	1246	(705)	1430	(837)	479	(300)	315	(272)
PFIT4LS	1932	(1095)	2147	(1261)	683	(432)	509	(416)
ROSENBR	70	(46)	96	(64)	52	(33)	38	(32)
S308	11	(11)	11	(11)	13	(13)	13	(13)
SINEVAL	238	(147)	268	(163)	106	(71)	76	(68)
SISSER	9	(9)	9	(9)	9	(9)	9	(9)
SNAIL	222	(148)	256	(190)	152	(96)	106	(95)
STRATEC	149	(85)	172	(103)	84	(70)	84	(69)
TOINTGOR	78	(51)	87	(56)	89	(73)	89	(73)
TOINTPSP	71	(42)	62	(41)	75	(54)	75	(54)
TOINTQOR	27	(25)	27	(25)	56	(37)	56	(37)
VIBRBEAM	134	(59)	109	(58)	125	(78)	114	(79)
YFITU	499	(262)	700	(377)	133	(85)	106	(83)
ZANGWIL2	2	(2)	2	(2)	2	(2)	2	(2)

fin du tableau

7.5.3 Illustration : calibration d'une loi de comportement élastoplastique.

Cette application est un test simple dans le domaine de la mécanique des solides [51, 52]. Le but est de calibrer les deux paramètres E et h de la loi de comportement d'un matériau élastoplastique

$$\sigma(\varepsilon) = \begin{cases} E\varepsilon & \text{si } \sigma < \sigma_0 \\ \sigma_0 + h(\varepsilon - \sigma_0/E) & \text{si } \sigma \geq \sigma_0 \end{cases} \tag{7.18}$$

où σ est la contrainte[1], ε le déplacement, E le module de Young, h le coefficient d'écrouissage et σ_0 la limite élastique. La géométrie de ce cas-test est représenté sur la figure 7.5. Il s'agit d'un triangle en état plan de contrainte. Sa partie inférieure est fixe et une force horizontale est appliquée sur le coin supérieur. Un maillage sommaire est utilisé (onze nœuds et six éléments) pour permettre d'effectuer des simulations rapides. Toutes les simulations numériques ont été effectuées avec *Lagamine*, un code éléments finis pour grandes déformations développé à l'Université de Liège (voir Charles *et al.* [15]).

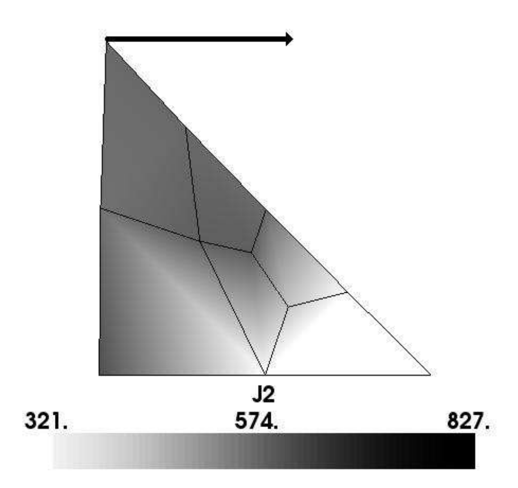


FIG. 7.5 – Géométrie et maillage du cas-test. Le triangle est encastré à sa base et une force horizontale de 18 N est appliquée au coin supérieur. La figure donne une mesure de l'état de contrainte après déformation dans le matériau, la contrainte équivalente de Von-Mises J_2 (en MPa).

Dans des applications réelles, le modèle doit être calibré par rapport à des données expérimentales. Dans cet exemple numérique toutefois, une *expérience*

[1] Au sens de la mécanique du solide déformable, *i.e.* une mesure de l'état de tension interne d'un solide (en anglais : *stress*). À ne pas confondre avec les contraintes en optimisation (en anglais : *constraints*).

jumelle est utilisée : une solution de référence est générée avec le modèle numérique lui-même[2]. Une simulation numérique donne la courbe force-déplacement et des points pseudo-expérimentaux sont générés avec

$$x^{\text{ref}} = \begin{bmatrix} E^{\text{ref}} & h^{\text{ref}} \end{bmatrix}^T. \tag{7.19}$$

Ces paramètres de références sont ensuite perturbés pour générer un point de départ $x^{(0)}$. L'algorithme est utilisé pour tenter de recouvrer la courbe de référence. Les valeurs numériques sont listées dans le tableau 7.7. La fonction objectif mesure l'écart entre les deux courbes au sens des moindres carrés

$$f(x) = \frac{1}{2} \left\| u(x) - u(x^{\text{ref}}) \right\|_2^2 \tag{7.20}$$

où u est le vecteur déplacement des noeuds. La matrice Jacobienne et le gradient sont calculés grâce à une technique de différentiation semi-analytique spécifique (voir Michaleris *et al.* [72]).

TAB. 7.7 – Valeurs numériques (en MPa) pour le point de départ $x^{(0)}$ et les paramètres de référence x^{ref}, *i.e.* le minimum global. Les deux dernières colonnes donnent les valeurs et les variations caractéristiques utilisées pour la mise à échelle (voir section 6.2.3). Noter que $\sigma_0 = 700$ MPa et n'est pas une variable à optimiser.

Paramètre	$x^{(0)}$	x^{ref}	$\widetilde{x}$	δx
E	220.000	200.000	200.000	60.000
h	270	300	300	90

Notre objectif est d'identifier les deux paramètres E et h de ce matériau. Puisque nous connaissons le minimum global du problème d'optimisation associé, nous introduisons une mesure de l'erreur relative à l'itération k

$$\varepsilon^{(k)} = \max \left\{ \left| \frac{E^{(k)} - E^{\text{ref}}}{E^{\text{ref}}} \right|, \left| \frac{h^{(k)} - h^{\text{ref}}}{h^{\text{ref}}} \right| \right\}. \tag{7.21}$$

Le tableau 7.8 donne le nombre d'itérations nécessaires pour approcher les paramètres de référence avec une erreur relative inférieure à 1% en utilisant l'approche BFGS inconditionnelle. Nous pouvons constater que la mise à jour du rayon de confiance de type R_1 ne parvient pas à approcher le minimum global à moins de 1% : l'algorithme converge en effet vers un minimum local dont l'erreur relative résiduelle est de $2,7\%$.

[2] Pour plus de précisons sur le principe de l'expérience jumelle, consulter la section 5.3.

TAB. 7.8 – Tableau comparatif du nombre d'itérations nécessaires pour atteindre 1% d'erreur relative pour différentes mises à jour du rayon de confiance. Les valeurs des paramètres de l'algorithme sont celles du tableau 6.1 et le rayon de confiance initial est égal à l'unité $\Delta^{(0)} = 1$. La mise à échelle du problème utilise les valeurs du tableau 7.7 (voir section 6.2.3).

Mise à jour du rayon de confiance	Nombre d'itérations
R_1	échec
R_2	16
Λ_1	9
Λ_2	11

La figure 7.6 montre l'évolution des itérés dans l'espace des variables de contrôle pour les mises à jour du rayon de confiance utilisant les fonctions R_2 et Λ_2. Rappelons que celles-ci sont identiques lorsque le rapport $\rho^{(k)}$ est inférieur à l'unité. Cette figure peut être utilisée pour illustrer l'interaction entre la mise à jour du rayon de confiance et celle de l'approximation du Hessien. Les trois premières itérations sont identiques pour les deux algorithmes. La troisième itération s'avère très réussie avec ($\rho^{(3)} = 1,6$) et le rayon est donc augmenté d'un facteur $2,92$ lorsque la fonction R_2 est utilisée et seulement $1,01$ pour Λ_2. En conséquence, deux points de test $\tilde{x}^{(4)}$ très différents sont obtenus, tous deux sont infructueux. L'itération 5 avec R_2 est dans la bonne direction (comparer avec le point 5 de la figure Λ_2) mais le rayon de confiance est trop grand et l'itération est, à nouveau, infructueuse. À cette étape-ci de l'algorithme, le Hessien a été pollué par deux points de tests plutôt lointains, menant ainsi l'algorithme vers un point $x^{(6)}$ qui n'est pas dans la direction du minimum global. Avec Λ_2, le point de test $\tilde{x}^{(4)}$ est également infructueux mais n'est pas aussi éloigné du point $x^{(3)}$, la mise à jour du Hessien n'en est que meilleure. L'itération 5 qui en résulte est réussie et $x^{(5)}$ se rapproche du minimum global.

La règle empirique de Byrd *et al.* (7.17) a également été utilisée sur ce problème illustratif. La figure 7.7 (à gauche) donne l'historique des itérations en utilisant Trust-BFGS-R_2 avec la règle (7.17). Nous pouvons constater que les premières itérations se contentent d'effectuer une recherche linéaire dans la direction $s^{(1)}$ qui est *grosso modo* la direction de plus grande pente. Ce comportement s'explique facilement, la condition (7.17) est nécessairement satisfaite lorsque $x^{(k)} = x^{(0)}$ et que l'itération est infructueuse : la matrice hessienne reste donc inchangée et égale à sa valeur initiale, généralement l'identité. Il est donc intéressant de désactiver ce « gardien » tant que $x^{(k)} = x^{(0)}$. La figure 7.7 (à droite)

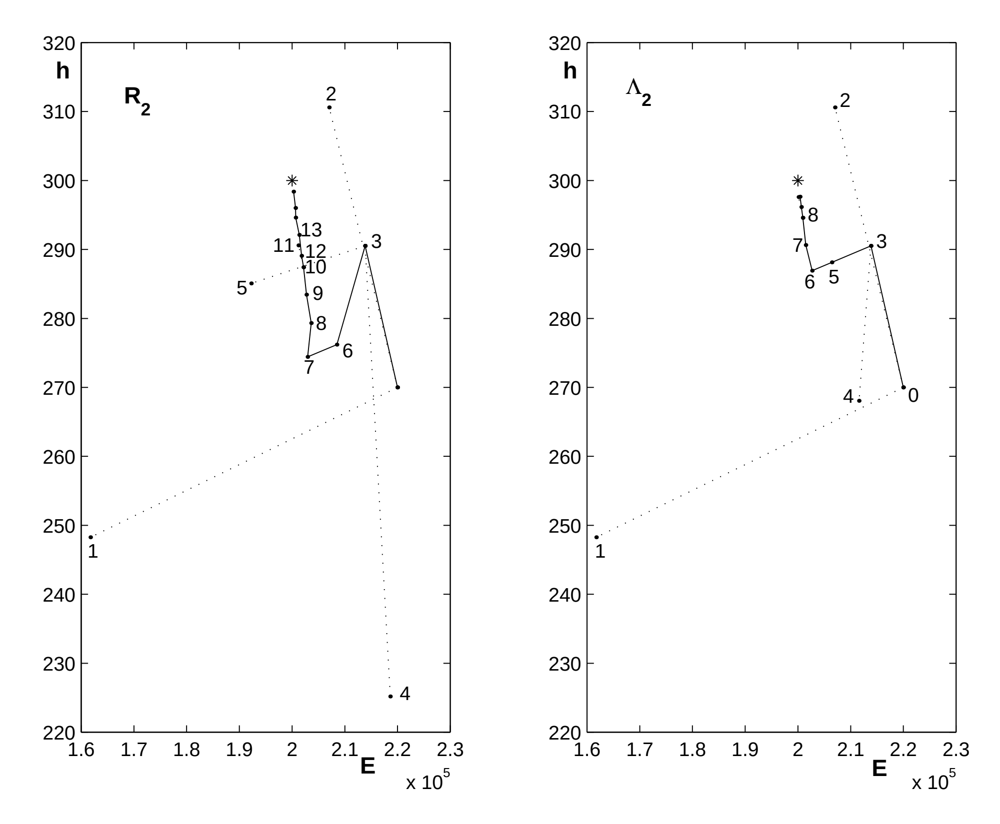


FIG. 7.6 – Historique des itérations pour le problème d'identification des paramètres d'une loi élastoplastique pour Trust-BFGS avec mise à jour quasi-Newton inconditionnelle. Les deux stratégies de mise à jour du rayon de confiance permettent à l'algorithme de converger vers le minimum global représenté par une astérisque. Les lignes pleines désignent les itérations réussies et les lignes pointillées les itérations infructueuses.

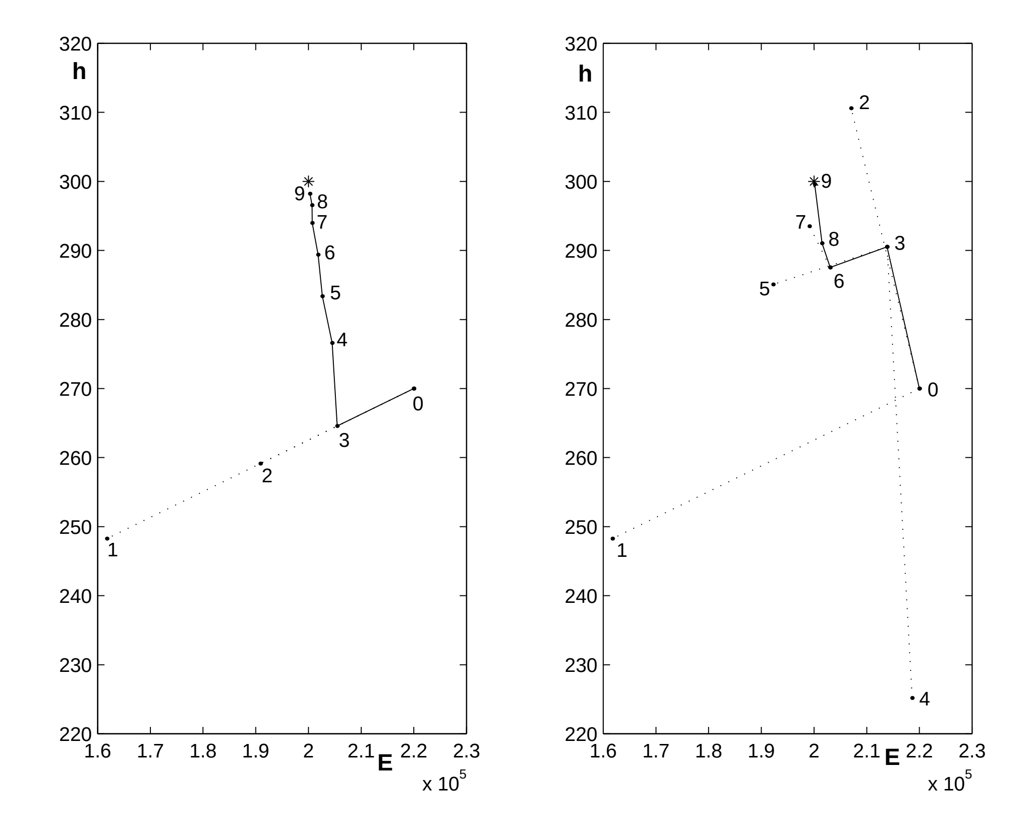


FIG. 7.7 – Historique des itérations pour le problème d'identification des paramètres d'une loi élastoplastique pour Trust-BFGS-R_2 en utilisant la règle empirique de Byrd *et al.* L'algorithme converge vers le minimum global représenté par une astérisque. Les lignes pleines désignent les itérations réussies et les lignes pointillées les itérations infructueuses. La figure de droite montre la trajectoire obtenue en désactivant la règle de Byrd *et al.* aussi longtemps que $x^{(k)} = x^{(0)}$.

présente l'historique des itérations. En comparant avec la figure 7.6 (à gauche), nous constatons que les cinq premières itérations sont les mêmes qu'en utilisant Trust-BFGS-R_2 avec mise à jour quasi-Newton inconditionnelle. Cependant, la condition (7.17) n'est pas satisfaite au point de test $x^{(5)}$ et la mise à jour du Hessien n'est dès lors pas effectuée[3]. Nous constatons que la trajectoire résultante est finalement plus proche de celle obtenue avec Λ_2 que de celle obtenue avec R_2.

[3]C'est par ailleurs la seule itération où cette condition est satisfaite.

7.6 Conclusion.

Dans les algorithmes utilisant ce type de globalisation, la mise à jour du rayon de la région de confiance est susceptible d'avoir une forte influence sur ses propriétés de convergence. Ce chapitre fournit une nouvelle stratégie de mise à jour du rayon de confiance basée sur l'idée que certaines itérations, d'apparence très réussies, sont en réalité *trop* réussies. C'est ce qui arrive lorsque la réduction de la fonction objectif s'avère significativement plus grande que celle qui était espérée après l'analyse de l'approximation locale. Dans ce cas, contrairement à l'habitude, nous suggérons de maintenir le rayon de confiance quasi constant.

Cette stratégie est très intuitive et largement applicable. La mise à jour auto-adaptative proposée conserve les propriétés générales de convergence de la globalisation par régions de confiance. Lorsque l'algorithme est proche de la convergence, la plupart des itérations sont très réussies et la région de confiance inopérante dans la minimisation de l'approximation locale. Le taux de convergence n'est donc pas affecté par la nouvelle règle de mise à jour du rayon.

Des expériences numériques conjuguant des algorithmes utilisant des approximations locales quadratiques de type quasi-Newton avec différentes stratégies de mise à jour du rayon de confiance montrent que la nouvelle approche améliore la vitesse de l'algorithme. En effet, cette règle permet d'éviter la pollution de l'approximation du Hessien avec des mises à jour de quasi-Newton imprécises. Malgré une légère détérioration de la robustesse, il apparaît que l'algorithme le plus efficace combine cette nouvelle stratégie avec une mise à jour inconditionnelle de l'approximation de la matrice hessienne par une règle de type quasi-Newton.

Troisième partie

Optimisation sous contraintes

Chapitre 8

Méthode SQP avec régions de confiance

Dans les chapitres précédents, nous avons décrit la forme générale des algorithmes avec régions de confiance. Le chapitre 4 a permis de présenter les résultats théoriques. En se basant sur ces derniers, le présent chapitre et les suivants envisagent la gestion de *contraintes* d'égalité ou d'inégalité, pour étendre l'algorithme de type quadratique séquentiel développé au chapitre 6 dans le cadre de l'optimisation non contrainte ou ne comprenant que des contraintes de bornes.

L'objet de ce chapitre est d'introduire les principes sous-tendant les algorithmes de programmation quadratique récursive ou *sequential quadratic programming* (SQP). Comme son nom l'indique, cette méthode procède par itérations sur des problèmes quadratiques approchant le problème d'optimisation original. Cette méthode est une des plus répandues et des plus efficaces dans le cadre de l'optimisation avec contraintes (*e.g.* [41, 46]). Nous nous attarderons en particulier sur l'utilisation d'une méthode SQP avec une méthode de globalisation par régions de confiance.

Notre but sera donc de traiter efficacement le problème d'optimisation nonlinéaire

$$\begin{array}{rll} \text{minimiser} & f(x), & \\ \text{s.c.} & c_j(x) = 0 & \text{pour } j \in \mathcal{E}, \\ & c_j(x) \leq 0 & \text{pour } j \in \mathcal{I} \end{array} \tag{8.1}$$

où $\mathcal{E}$ et $\mathcal{I}$ sont, respectivement, les ensembles disjoints des indices des contraintes d'égalité et d'inégalité. Les fonctions f et c_j sont supposées continûment dérivables. Nous supposerons également l'absence de dégénérescence géométrique entre les différentes contraintes, ce qui signifie que l'hypothèse de *qualification des contraintes* est satisfaite (voir section 1.2).

8.1 Principe de base

Les méthodes SQP sont généralement introduites en considérant, dans en premier temps, le problème dépourvu de contrainte d'inégalité

$$\begin{array}{rl} \text{minimiser} & f(x), \\ \text{s.c.} & c_j(x) = 0 \quad \text{pour } j \in \mathcal{E} \end{array} \tag{8.2}$$

dont le Lagrangien s'écrit

$$\mathcal{L}(x,y) = f(x) - \sum_{j \in \mathcal{E}} y_j c_j(x) \tag{8.3}$$

où y est un ensemble de *multiplicateurs de Lagrange*. La méthode la plus simple pour résoudre ce problème est certainement de résoudre les conditions d'optimalité de Karush–Khun–Tucker (1.8).

$$\nabla_x \mathcal{L}(x^*, y^*) = 0 \qquad \text{et} \qquad c(x^*) = 0 \tag{8.4}$$

par la méthode itérative de Newton.

Si nous disposons d'estimations $x^{(k)}$ et $y^{(k)}$ des valeurs critiques x^* et y^*, l'itération de Newton s'écrit

$$\begin{pmatrix} x^{(k+1)} \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} + \begin{pmatrix} s_x^{(k)} \\ s_y^{(k)} \end{pmatrix} \tag{8.5}$$

et les incréments $s_x^{(k)}$ et $s_y^{(k)}$ s'obtiennent en inversant le système

$$\begin{pmatrix} \nabla_{xx} \mathcal{L}(x^{(k)}, y^{(k)}) & -G^{(k)} \\ G^{(k)T} & 0 \end{pmatrix} \begin{pmatrix} s_x^{(k)} \\ s_y^{(k)} \end{pmatrix} = - \begin{pmatrix} \nabla_x \mathcal{L}(x^{(k)}, y^{(k)}) \\ c^{(k)} \end{pmatrix} \tag{8.6}$$

où $c^{(k)} = c(x^{(k)})$ et

$$G^{(k)} = \left[\nabla_x c_j(x^{(k)}) \right]_{j \in \mathcal{E}}. \tag{8.7}$$

Il est aisé de vérifier que (8.6) peut également être écrit sous la forme

$$\begin{pmatrix} \nabla_{xx} \mathcal{L}(x^{(k)}, y^{(k)}) & -G^{(k)} \\ G^{(k)T} & 0 \end{pmatrix} \begin{pmatrix} s_x^{(k)} \\ y^{(k+1)} \end{pmatrix} = - \begin{pmatrix} g^{(k)} \\ c^{(k)} \end{pmatrix}. \tag{8.8}$$

Il y a une autre façon d'obtenir le processus itératif (8.8). Examinons le problème quadratique suivant

$$\begin{array}{rl} \text{minimiser} & \frac{1}{2} s^T H^{(k)} s + g^{(k)T} s \\ \text{s.c.} & G^{(k)T} s + c^{(k)} = 0 \end{array} \tag{8.9}$$

dont le Lagrangien s'écrit

$$\mathcal{L}^{(k)}(s,y) = \frac{1}{2}s^T H^{(k)} s + g^{(k)T} s - y^T (G^{(k)T} s + c^{(k)}) \tag{8.10}$$

où y est le vecteur des multiplicateurs de Lagrange relatif aux contraintes d'égalité. Les conditions de stationnarité du Lagrangien pour obtenir les points critiques $s_x^{(k)}$ et $y^{(k+1)}$ s'écrivent

$$H^{(k)} s_x^{(k)} + g^{(k)} - G^{(k)} y^{(k+1)} = 0 \tag{8.11}$$

$$G^{(k)T} s_x^{(k)} + c^{(k)} = 0 \tag{8.12}$$

qui est parfaitement équivalent au système (8.8) si $H^{(k)} = \nabla_{xx}\mathcal{L}(x^{(k)}, y^{(k)})$. Le pas de progression $s_x^{(k)}$ est donc aussi celui qui minimise le problème (8.9) et $y^{(k+1)}$ est le vecteur des multiplicateurs de Lagrange correspondant. C'est à cette équivalence que la méthode doit son nom de programmation quadratique séquentielle.

Notons que la littérature regorge de méthodes pour résoudre les problèmes du type (8.9) (voir par exemple [4, 11, 29, 31, 42, 56, 80]). La grande majorité des méthodes SQP proposent de remplacer $H^{(k)}$ par une approximation du Hessien du Lagrangien. Une des raisons principales du grand intérêt suscité par ces méthodes réside dans son potentiel de convergence rapide. Celui-ci est établi par le théorème suivant [20].

Théorème 8.1 *Supposons que les dérivées secondes de $f(x)$ et $c_i(x)$ existent et soient continues au sens de Lipschitz dans un voisinage Ω du point critique du premier ordre (x^*, y^*) et que la matrice apparaissant dans le membre de gauche de* (8.6) *soit non singulière. Soit la suite $\{x^{(k)}\}$ générée par l'itération* (8.5) *avec $s_x^{(k)}$ solution de* (8.9).

- *Soit une suite quelconque $\{y^{(k)}\}$ convergeant vers y^*. Alors, il existe un voisinage $\mathcal{X} \subset \Omega$ de x^* tel que la suite $\{x^{(k)}\}$ converge de manière Q-superlinéaire vers x^* à partir de n'importe quel point de départ $x^{(0)}$ de $\mathcal{X}$. Si $\|y^{(k)} - y^*\| = \mathcal{O}(\|x^{(k)} - x^*\|)$, la convergence est quadratique.*
- *Soit la suite $\{y^{(k)}\}$ des multiplicateurs de Lagrange du problème* (8.9). *Alors, il existe un voisinage $\mathcal{X} \subset \Omega$ et un voisinage $\mathcal{Y}$ de y^* tels que la suite $\{(x^{(k)}, y^{(k)})\}$ converge de manière Q-quadratique vers (x^*, y^*) à partir de n'importe quel point de départ $(x^{(0)}, y^{(0)})$ de $\mathcal{X} \times \mathcal{Y}$.*

Notons qu'au vu de ce théorème, il n'est pas nécessaire de prendre $y^{(k+1)}$ comme le vecteur des multiplicateurs de Lagrange du problème (8.9) pour obtenir un taux de convergence Q-superlinéaire. En pratique, un estimateur au sens des moindres carrés est souvent utilisé

$$y^{(k)} = \arg\min_y \|g^{(k)} - G^{(k)} y\|^2. \tag{8.13}$$

Si $G(x^*)$ est non-singulière, on peut montrer que (voir [20])

$$\|y^{(k)} - y^*\| = o\,(\|x^{(k)} - x^*\|) \tag{8.14}$$

et, de ce fait, que la suite $\{x^{(k)}\}$ converge de manière Q-quadratique.

La prise en compte de contraintes d'inégalité se fait généralement par l'utilisation d'une stratégie de contraintes actives qui tente de déterminer par avance quelles seront les contraintes actives à l'optimum. Lorsque ces contraintes sont connues, le problème peut être résolu comme s'il ne faisait intervenir que des contraintes d'égalité. De plus amples détails sur ces stratégies de contraintes actives peuvent être trouvées dans [20, 31, 80].

8.2 Fonction de mérite et globalisation

Tout comme les méthodes développées dans le cadre de l'optimisation non-contrainte, les méthodes SQP constituent des approximations locales qui doivent être utilisées en conjonction avec une technique de globalisation pour en assurer la convergence. Pour guider le processus conduisant à diminuer la fonction objectif et la violation éventuelle des contraintes, on introduit une *fonction de mérite* qui joue le rôle de la fonction objectif en optimisation non-contrainte. Notons que, récemment, certains auteurs ont développé des méthodes — dites avec filtres — n'utilisant pas de fonction de mérite. Le détail de ces méthodes sort du cadre de ce travail mais le lecteur intéressé pourra consulter, entre autres, les travaux de Fletcher et Leyffer [33] et de Fletcher *et al.* [32].

Bon nombre de fonctions de mérite ont été utilisées dans le cadre des méthodes de type SQP. Nous nous focaliserons ici sur la fonction de mérite non-différentiable ℓ_1. Pour le problème (8.1), celle-ci est définie par

$$\Psi(x,\sigma) = f(x) + \sigma \sum_{j\in\mathcal{E}} |c_j(x)| + \sigma \sum_{j\in\mathcal{I}} \max[0, c_j(x)] \tag{8.15}$$

où $\sigma > 0$ est un paramètre de pénalité. Cette fonction n'est pas différentiable partout ; en particulier, aux points d'annulation d'une ou plusieurs composantes $c_j(x)$, le gradient n'est pas défini. Elle possède cependant une propriété remarquable énoncée par le théorème suivant [20].

Théorème 8.2 *Soient $f(x)$ et $c_j(x)$ des fonctions deux fois continûment dérivables. Supposons que x^* et y^* sont telles que x^* est admissible pour le problème* (8.1) *et que*

$$\sigma \geq \|y^*\|_\infty = \max_{j\in\mathcal{E}\cup\mathcal{I}} |y_j^*|. \tag{8.16}$$

Dans ce cas, si x^* et y^* satisfont aux conditions suffisantes d'optimalité du premier ordre pour (8.1)*, x^* satisfait aussi aux conditions suffisantes du premier ordre d'un minimum local de $\Psi(x,\sigma)$.*

Ce théorème implique que si nous pouvons trouver un point admissible qui satisfait les conditions suffisantes du second ordre pour la fonction de mérite, il sera solution du problème non-linéaire associé (8.1). Cette propriété fait de la fonction de mérite ℓ_1 une fonction de mérite *exacte*, par opposition avec d'autres fonctions de mérite qui ne permettent d'obtenir qu'une solution approximative (voir [80, 20]). Sous certaines conditions, le problème non-linéaire contraint (8.1) peut donc être transformé en un problème non-contraint, bien que non-différentiable. Le théorème suivant nous permet de mesurer l'importance de la condition (8.16).

Théorème 8.3 *Soient $f(x)$ et $c_j(x)$ des fonctions deux fois continûment dérivables. Supposons que x^* est un point critique du premier ordre du problème* (8.1) *et que y^* sont les multiplicateurs de Lagrange correspondants. Si*

$$\sigma < \|y^*\|_\infty, \tag{8.17}$$

alors x^* n'est pas un minimum local de $\Psi(x,\sigma)$.

Nous constatons donc que pour des valeurs de σ plus grande que $\|y^*\|_\infty$, la minimimisation de la fonction de mérite nous amène très sûrement à minimiser le problème non-linéaire (8.1) qui peut donc être remplacé par

$$\text{minimiser } \Psi(x,\sigma). \tag{8.18}$$

Toutefois, le problème de minimisation non-contrainte de la fonction de pénalité (8.15) est plus difficile à résoudre numériquement lorsque σ croît. En effet, celui-ci devient mal conditionné pour de grandes valeurs de σ (voir figure 8.1). L'idéal est donc d'adopter une valeur de σ légèrement supérieure à la borne $\|y^*\|_\infty$ dont la valeur est malheureusement inconnue. Certains algorithmes prévoient une adaptation du paramètre σ en cours de calcul afin de coller au mieux à cette condition.

Exemple 8.1 *Considérons le problème proposé par Powell [84]*

$$\begin{array}{rl} \textit{minimiser} & f(x) = 2(x_1^2+x_2^2-1) - x_1 \\ \textit{s.c.} & c(x) = x_1^2+x_2^2-1 = 0 \end{array} \tag{8.19}$$

dont la solution est $(1,0)$ avec $y^* = 3/2$ comme multiplicateur de Lagrange. La fonction de mérite ℓ_1 pour ce problème s'écrit

$$\Psi(x,\sigma) = 2(x_1^2+x_2^2-1) - x_1 + \sigma\,|x_1^2+x_2^2-1| \tag{8.20}$$

et est représentée sur la figure 8.1 pour différentes valeurs de σ. *Nous pouvons constater que, pour une valeur* $\sigma < y^*$, *la minimisation de la fonction de mérite* (8.20) *ne conduit pas au minimum du problème contraint* (8.19). *En revanche, lorsque* $\sigma \geq y^*$, *les deux minima sont identiques. Remarquons également que la croissance du paramètre* σ *a tendance à rendre de plus en plus escarpée la « vallée » autour de la contrainte, ce qui détériore le conditionnement de la minimisation de la fonction de mérite.*

L'utilisation d'une fonction de mérite permet d'introduire les deux techniques de globalisation utilisées dans ce travail : la *recherche linéaire* et les *régions de confiance*.

Les méthode SQP avec *recherche linéaire* cherchent à améliorer une estimation $(x^{(k)}, y^{(k)})$ de la solution de (8.2) en calculant des corrections $(s_x^{(k)}, s_y^{(k)})$ par résolution du problème quadratique (8.9). L'itéré suivant sera

$$\begin{pmatrix} x^{(k+1)} \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} + \xi^{(k)} \begin{pmatrix} s_x^{(k)} \\ s_y^{(k)} \end{pmatrix} \tag{8.21}$$

où $\xi^{(k)}$ est un pas de progression calculé de façon à assurer, de manière analogue à ce qui est fait pour la fonction objectif dans un problème non-contraint, une certaine décroissance de la fonction de mérite dans la direction $s_x^{(k)}$. Il convient donc de résoudre, éventuellement de façon approchée, le problème unidimensionnel

$$\xi^{(k)} \in \arg\min_{\xi} \Psi(x^{(k)} + \xi s_x^{(k)}, \sigma) \tag{8.22}$$

qui permet de guider le processus itératif vers une amélioration de la solution à chaque étape.

Exemple 8.2 *À titre d'exemple, construisons, pour le problème* (8.19), *le sous-problème quadratique* (8.9) *correspondant au point* $x^{(k)} = (1/4, 1/4)$ *et utilisant la valeur optimale pour le multiplicateur de Lagrange* $y^{(k)} = 3/2$. *La matrice hessienne du Lagrangien s'écrit*

$$H^{(k)} = \nabla_{xx} f(x^{(k)}) - y^{(k)} \nabla_{xx} c(x^{(k)}) = 4I - \frac{3}{2} 2I = I, \tag{8.23}$$

tandis que le gradient de la fonction objectif et la matrice des gradients des contraintes sont données par

$$g(x) = \begin{pmatrix} 4x_1 - 1 \\ 4x_2 \end{pmatrix} \qquad \text{et} \qquad G(x) = \begin{pmatrix} 4x_1 \\ 4x_2 \end{pmatrix}. \tag{8.24}$$

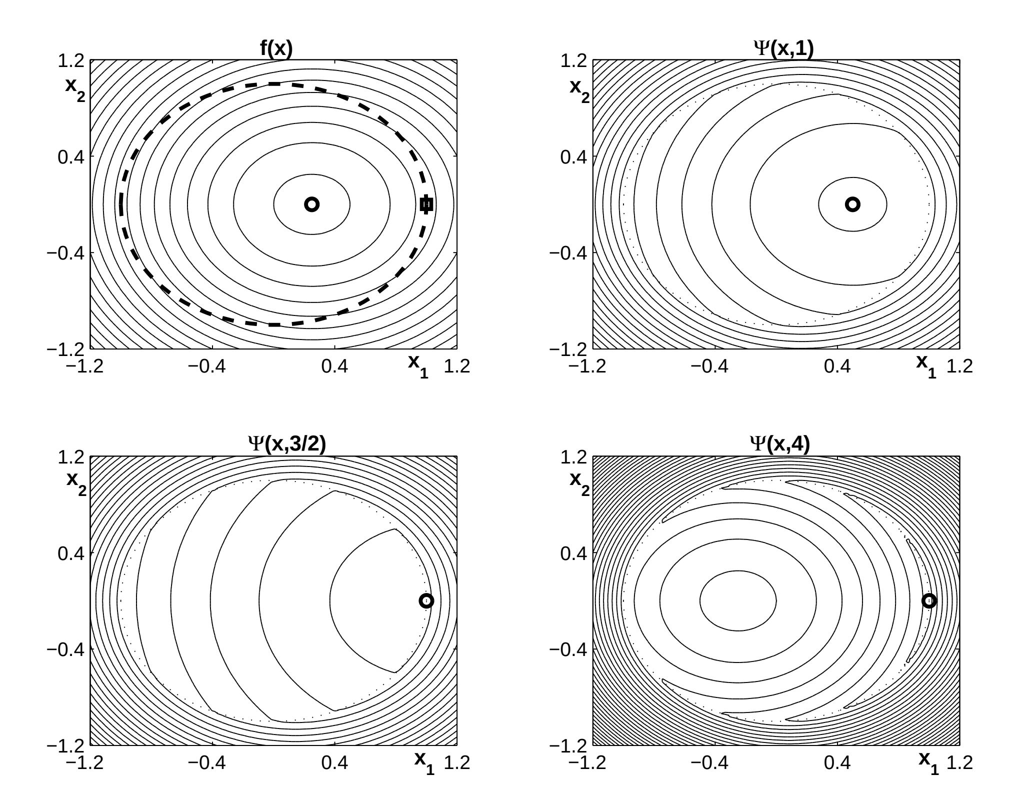


FIG. 8.1 – Le problème (8.19) est représenté sur la figure supérieure gauche : les iso-valeurs de la fonction objectif, la contrainte d'égalité (trait épais discontinu), le minimum global (carré) et le minimum global non-contraint (cercle). Les trois autres figures représentent la fonction de mérite ℓ_1 pour différentes valeurs de σ : le cercle indique la position du minimum de Ψ. Nous constatons que le minimum non-contraint de Ψ ne coïncide pas avec le minimum contraint du problème (8.19) si $\sigma < y^* = 3/2$, contrairement aux cas où $\sigma \geq 3/2$. Nous pouvons également remarquer que l'escarpement de la « vallée » autour de la contrainte s'accroît avec le paramètre σ : ceci peut causer des problèmes de conditionnement lors de la minimisation de la fonction de mérite.

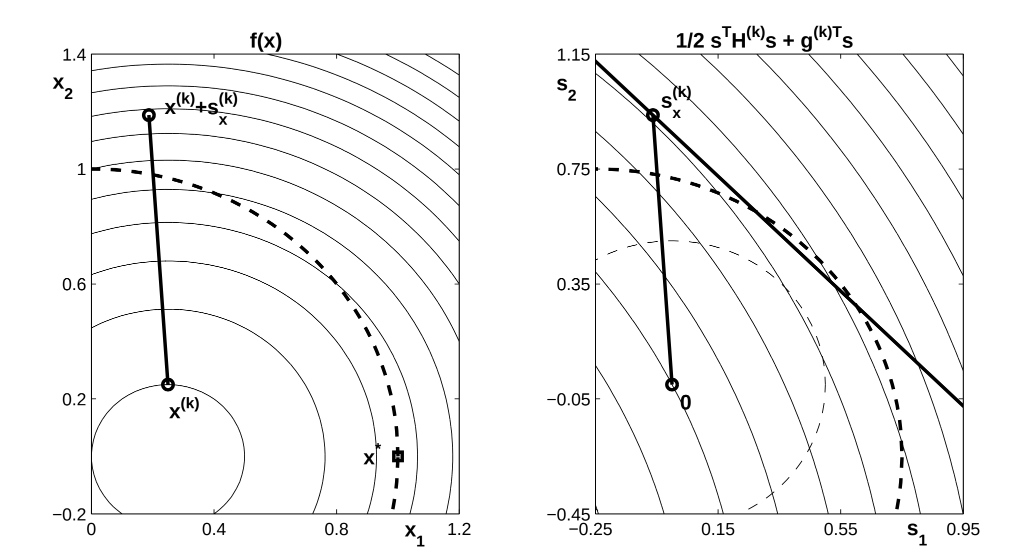


FIG. 8.2 – Mécanisme de base d'une itération SQP. Le problème (8.19) est représenté sur la figure de gauche : les iso-valeurs de la fonction objectif, la contrainte d'égalité (trait épais discontinu), le minimum global, le point courant et le pas de progression. Le problème (8.25) est représenté sur la figure de droite : les iso-valeurs de la fonction quadratique, la contrainte non-linéaire (trait épais discontinu), la contrainte linéarisée (trait épais continu) et le pas de progression obtenu. Le cercle en trait fin discontinu est une région de confiance de rayon $\Delta^{(k)} = 1/2$: nous constatons qu'il est impossible dans ce cas d'être à la fois à l'intérieur de la région de confiance et de satisfaire à la contrainte d'égalité linéarisée.

Le sous-problème quadratique (8.9) *correspondant s'écrit*

$$\begin{aligned} \textit{minimiser} \quad & 1/2\,(s_1^2+s_2^2)+2\,s_1+s_2 \\ \textit{s.c.} \quad & s_1+s_2-7/8=0 \end{aligned} \qquad (8.25)$$

dont la solution est $s_x^{(k)} = (-1/16, 15/16)$. *Le problème* (8.19) *et le sous-problème quadratique correspondant* (8.25) *sont représentés à la figure 8.2. Nous constatons que la courbure de la fonction utilisée comme nouvelle fonction objectif a été modifiée en fonction de la courbure de la contrainte et que cette dernière a été remplacée par une contrainte linéaire.*

L'approche par *régions de confiance* utilise le sous-problème (8.9) tout en introduisant une contrainte de confinement supplémentaire

$$\begin{aligned} \text{minimiser} \quad & \tfrac{1}{2}s^T H^{(k)} s + g^{(k)T} s \\ \text{s.c.} \quad & G^{(k)T} s + c^{(k)} = 0 \\ \text{et} \quad & \|s\|_k \le \Delta^{(k)} \end{aligned} \qquad (8.26)$$

avec une norme $\|\cdot\|_k$ et un rayon de confiance $\Delta^{(k)}$ donnés. Avec une solution $s_x^{(k)}$ (éventuellement approchée) au problème (8.26), un *point de test* peut être construit

$$\tilde{x}^{(k)} = x^{(k)} + s_x^{(k)}. \tag{8.27}$$

La valeur de la *fonction de mérite* à ce point de test est alors calculée $\Psi(\tilde{x}^{(k+1)}, \sigma)$ et comparée à celle escomptée à partir d'une *approximation locale*

$$\Psi_m(x^{(k+1)} + s, \sigma). \tag{8.28}$$

Si l'adéquation entre l'approximation locale et la fonction originale est satisfaisante, le rayon de confiance est accru. Dans le cas contraire, celui-ci est réduit. Le schéma de l'algorithme est tout à fait analogue à celui développé à la section 2.2 pour les problèmes non-contraints mais c'est la fonction de mérite qui tient lieu de fonction objectif. Mais c'est sans compter sur une différence majeure avec la globalisation par recherche linéaire : il n'y a *a priori* aucune raison que le problème (8.26) possède une solution. La figure 8.2 présente un exemple de problème sans solution à deux dimensions. Une autre question problématique est celle de l'approximation locale $\Psi_m(x^{(k+1)} + s, \sigma)$: comment doit elle-être choisie ? Les algorithmes SQP utilisant une globalisation par régions de confiance doivent tenir compte de ces questions. Plusieurs mécanismes ont été proposés pour pallier ce problème : le lecteur intéressé est invité à consulter l'ouvrage de Conn *et al.* [20].

Exemple 8.3 *La figure 8.2 illustre une de ces situations sur le problème* (8.19). *Adoptons une norme euclidienne pour la région de confiance et construisons le sous-problème quadratique* (8.26) *correspondant au point* $x^{(k)} = (1/4, 1/4)$ *tout en utilisant la valeur optimale pour le multiplicateur de Lagrange* $y^{(k)} = 3/2$

$$\begin{aligned} \textit{minimiser} \quad & 1/2\,(s_1^2 + s_2^2) + 2s_1 + s_2 \\ \textit{s.c.} \quad & s_1 + s_2 - 7/8 = 0 \\ \textit{et} \quad & \sqrt{s_1^2 + s_2^2} \leq \Delta^{(k)}. \end{aligned} \tag{8.29}$$

Le problème (8.19) *et le sous-problème quadratique correspondant* (8.29) *sont représentés sur la figure 8.2. Nous pouvons constater que, pour une valeur du rayon de confiance d'une demi-unité, le sous-problème quadratique ne possède pas de solution et que la stratégie habituelle de diminution du rayon de la région de confiance ne fait qu'exacerber le problème.*

Pour éviter ce genre de configuration, le problème (8.29) *peut être remplacé par la minimisation d'une approximation locale de la fonction de mérite* (8.20)

$$\Psi_m(s, \sigma) = \Psi(x^{(k)}, \sigma) + \frac{1}{2}(s_1^2 + s_2^2) + 2s_1 + s_2 + \sigma \left| s_1 + s_2^2 - \frac{7}{8} \right| \tag{8.30}$$

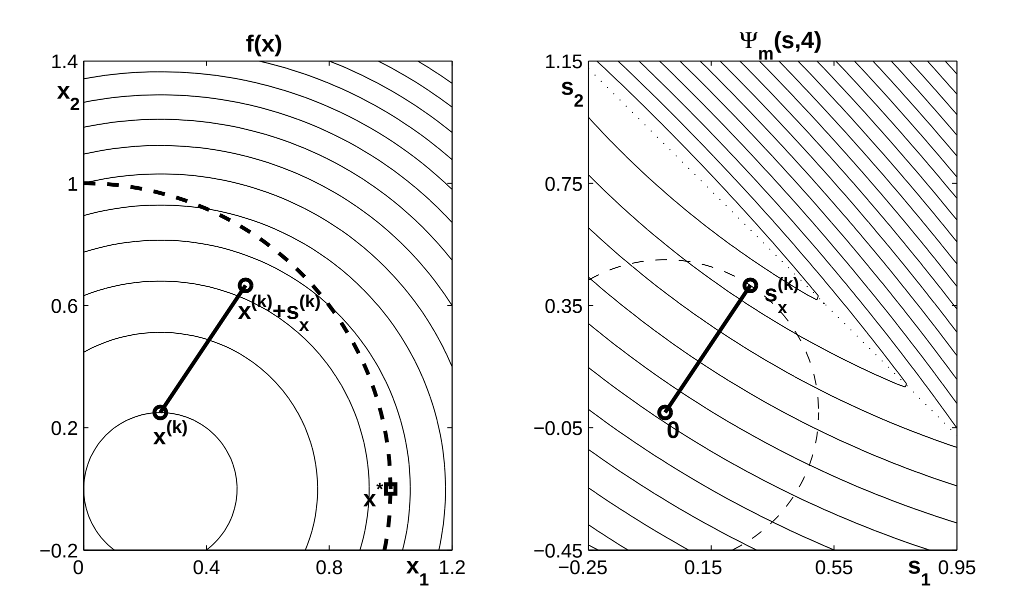


FIG. 8.3 – Exemple d'une itération SQP avec région de confiance. Le problème (8.19) est représenté sur la figure de gauche : les iso-valeurs de la fonction objectif, la contrainte d'égalité (trait épais discontinu), le minimum global, le point courant et le pas de progression. L'approximation locale (8.30) de la fonction de mérite est représentée sur la figure de droite ainsi que le pas de progression obtenu. Le cercle en trait fin discontinu est la région de confiance de rayon $\Delta^{(k)} = 1/2$: nous constatons que $s_x^{(k)}$ est le pas qui, au sein de la région de confiance, mène à un point qui établit un compromis entre les minimisations de la fonction objectif et de l'éloignement vis-à-vis de la ligne de discontinuité du gradient. Celle-ci correspond à la contrainte d'égalité linéaire du problème (8.29).

soumise à la seule contrainte de confinement

$$\sqrt{s_1^2 + s_2^2} \leq \Delta^{(k)}. \tag{8.31}$$

La solution $s_x^{(k)}$ de ce problème local n'est évidemment pas identique à celle du problème (8.25) *mais elle constitue un compromis, au sein de la région de confiance, entre la minimisation de la fonction objectif et la diminution de l'amplitude de la violation de la contrainte d'égalité. Ce problème est illustré sur la figure 8.3 pour $\sigma = 4$ et $\Delta^{(k)} = 1/2$.*

8.3 Effet Maratos et correction du second ordre

Bien qu'utile pour assurer la convergence globale, l'utilisation d'une fonction de mérite a des effets non-négligeables sur la vitesse de convergence. Il peut arriver que le pas de progression $s_x^{(k)}$ se voit rejeté ou considérablement diminué par le processus de globalisation. Or, un pas de progression sensiblement plus petit que $s_x^{(k)}$ freine l'algorithme : la vitesse de convergence augurée par le théorème 8.1 n'est en effet atteinte que pour le pas $s_x^{(k)}$, solution de (8.9). Ce ralentissement est appelé l'*effet Maratos* [69]. L'effet Maratos apparaît lorsque la courbure des contraintes n'est pas bien appréhendée par le sous-problème quadratique (8.9) et qu'un pas unitaire est trop grand pour que la contrainte linéarisée $c^{(k)} + G^{(k)T}s$ soit une approximation acceptable de la contrainte non-linéaire $c(x^{(k)} + s)$.

Exemple 8.4 *Nous allons illustrer l'effet Maratos à partir de l'exemple* (8.19). *Effectuons une itération de type SQP partant du point admissible*

$$x^{(k)} = (\cos\theta, \sin\theta) \tag{8.32}$$

avec $\theta > 0$ *et utilisant la valeur optimale pour le multiplicateur de Lagrange* $y^{(k)} = 3/2$*. Tenant compte de l'expression de la matrice hessienne du Lagrangien* (8.23) *et des gradients de la fonction objectif et de la contrainte* (8.24)*, le sous-problème quadratique* (8.9) *correspondant est*

$$\begin{array}{rl} \textit{minimiser} & \frac{1}{2}(s_1^2 + s_2^2) + (4\cos\theta - 1)\, s_1 + 4\sin\theta\, s_2 \\ \textit{s.c.} & 2\cos\theta\, s_1 + 2\sin\theta\, s_2 = 0. \end{array} \tag{8.33}$$

Sa solution est $s_x^{(k)} = (\sin^2\theta, -\cos\theta\sin\theta)$*. Cependant, on peut constater sur l'expression de la fonction de mérite (en utilisant la contrainte non-linéaire réelle) le long de la direction* $s_x^{(k)}$

$$\Psi(x^{(k)} + \xi s_x^{(k)}, \sigma) = (2+\sigma)\sin^2\theta\,\xi^2 - \sin^2\theta\,\xi - \cos\theta \tag{8.34}$$

que

$$\Psi(x^{(k)} + s_x^{(k)}, \sigma) - \Psi(x^{(k)}, \sigma) = (1+\sigma)\sin^2\theta > 0. \tag{8.35}$$

Le pas $s_x^{(k)}$ *susceptible de fournir une convergence rapide sera donc irrémédiablement rejeté aussi bien par une recherche linéaire que par une approche par régions de confiance. Le minimum unidimensionnel dans la direction* $s_x^{(k)}$ *est obtenu pour la valeur*

$$\xi^{(k)} = \frac{1}{2(2+\sigma)} \tag{8.36}$$

dont nous pouvons constater qu'elle décroît avec la croissance du paramètre de pénalité σ *et que sa valeur est largement inférieure à l'unité. La figure 8.4 présente la fonction objectif et la fonction de mérite (*$\sigma = 4$*).*

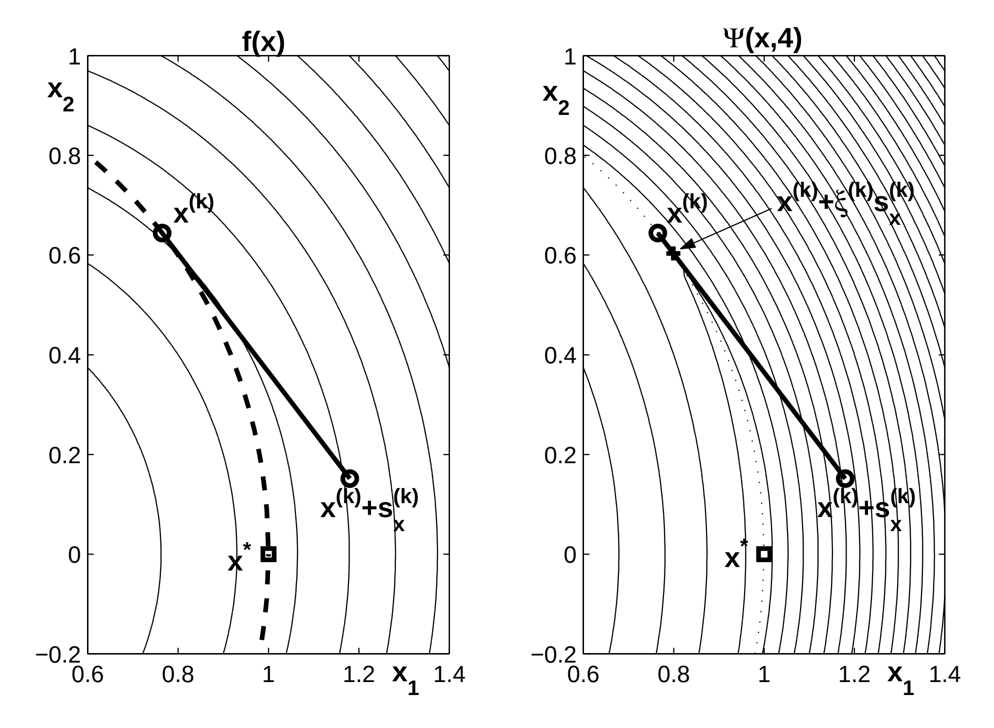


FIG. 8.4 – Illustration de l'effet Maratos pour le problème (8.19). La figure de gauche représente la fonction objectif $f(x)$ et la ligne discontinue représente la contrainte d'égalité $c(x) = 0$. La figure de droite représente la fonction de mérite pour $\sigma = 4$. Nous pouvons constater que $\Psi(x^{(k)} + s_x^{(k)}, 4) > \Psi(x^{(k)}, 4)$ et que le minimum de Ψ selon la direction $s_x^{(k)}$ (c'est-à-dire $x^{(k)} + \xi^{(k)} s_x^{(k)}$) conduit à restreindre fortement la progression vers x^*.

Pour contrecarrer l'effet Maratos, une des techniques utilisées est celle de la *correction du second ordre*. Il s'agit d'un pas de correction $s_c^{(k)}$ qui est appliqué au point $x^{(k)} + s_x^{(k)}$ de façon à ramener la valeur des contraintes (actives) à une valeur négligeable. En d'autre mots, puisque l'ordre de grandeur des contraintes au point $x^{(k)} + s_x^{(k)}$ est

$$c(x^{(k)} + s_x^{(k)}) = O(\|s_x^{(k)}\|^2), \tag{8.37}$$

la correction $s_c^{(k)}$ doit être telle que la valeur de la contrainte devienne négligeable devant cet ordre de grandeur antérieur

$$c(x^{(k)} + s_x^{(k)} + s_c^{(k)}) = o(\|s_x^{(k)}\|^2). \tag{8.38}$$

Cependant, il est important qu'une telle correction n'altère pas exagérément le pas original dont on cherche à préserver les propriétés de convergence rapide. Il convient donc également que la correction soit négligeable par rapport à $s_x^{(k)}$

$$s_c^{(k)} = o(\|s_x^{(k)}\|). \tag{8.39}$$

Tout vecteur $s_c^{(k)}$ satisfaisant aux relations (8.38) et (8.39) est une correction du second ordre.

Comment dès lors choisir $s_c^{(k)}$? Il existe plusieurs variantes mais la plus simple est certainement d'utiliser l'approximation de Taylor

$$c(x^{(k)} + s_x^{(k)} + s_c^{(k)}) = c(x^{(k)} + s_x^{(k)}) + G^T(x^{(k)} + s_x^{(k)})\, s_c^{(k)} + O(\|s_c^{(k)}\|^2) \tag{8.40}$$

qui peut être légèrement modifiée comme suit

$$\begin{aligned} c(x^{(k)} + s_x^{(k)} + s_c^{(k)}) \quad = \quad & c(x^{(k)} + s_x^{(k)}) + G^{(k)T} s_c^{(k)} \\ & + O\left[\|s_c^{(k)}\| \max\left(\|s_x^{(k)}\|, \|s_c^{(k)}\|\right)\right] \end{aligned} \tag{8.41}$$

en utilisant

$$G^T(x^{(k)} + s_x^{(k)}) = G^{(k)T} + O(\|s_x^{(k)}\|). \tag{8.42}$$

Si nous choisissons $s_c^{(k)}$ tel que

$$G^{(k)T} s_c^{(k)} + c(x^{(k)} + s_x^{(k)}) = 0, \tag{8.43}$$

nous obtenons de (8.41) que

$$c(x^{(k)} + s_x^{(k)} + s_c^{(k)}) = O\left[\|s_c^{(k)}\| \max\left(\|s_x^{(k)}\|, \|s_c^{(k)}\|\right)\right]. \tag{8.44}$$

Bien entendu, le système (8.43) de m équations à n inconnues peut être incompatible ou posséder une infinité de solutions. C'est pourquoi ce problème est généralement résolu au sens d'une certaine norme, le plus souvent $s_c^{(k)}$ est la solution de norme ℓ_2 minimum

$$s_c^{(k)} = \arg\min_s \left\| G^{(k)T} s + c(x^{(k)} + s_x^{(k)}) \right\|_2 \tag{8.45}$$

qui peut être calculée au moyen d'une décomposition QR de la matrice

$$G^{(k)} = Q^{(k)} \begin{pmatrix} R^{(k)} \\ 0 \end{pmatrix} = \begin{pmatrix} Q_1^{(k)} & Q_2^{(k)} \end{pmatrix} \begin{pmatrix} R^{(k)} \\ 0 \end{pmatrix} \tag{8.46}$$

où $Q^{(k)}$ est une matrice orthogonale carrée d'ordre n et $R^{(k)}$ une matrice carrée triangulaire supérieure d'ordre m. Les matrices $Q_1^{(k)}$ et $Q_2^{(k)}$ sont, respectivement, constituées des m premières colonnes de $Q^{(k)}$ et des $n-m$ colonnes restantes. Le problème (8.45) peut donc s'écrire

$$s_c^{(k)} = \arg\min_s \left\| R^{(k)T} Q_1^{(k)T} s + c(x^{(k)} + s_x^{(k)}) \right\|_2 \tag{8.47}$$

dont la solution est donnée par

$$s_c^{(k)} = Q^{(k)} \begin{pmatrix} -(R^{(k)})^{-1} c(x^{(k)} + s_x^{(k)}) \\ 0 \end{pmatrix} \tag{8.48}$$

En utilisant (8.37), nous pouvons déduire facilement l'ordre de grandeur de la correction

$$s_c^{(k)} = O(\|c^{(k)}\|) = O(\|s_x^{(k)}\|^2), \tag{8.49}$$

ce qui satisfait à la condition (8.39) et, en remplaçant dans (8.44), nous obtenons

$$c(x^{(k)} + s^{(k)} + s_c^{(k)}) = O(\|s_x^{(k)}\|^3). \tag{8.50}$$

et la condition (8.38) est également satisfaite. La correction $s_c^{(k)}$ obtenue en résolvant le système (8.43) est donc bien une correction du second ordre parfaitement valide. Notons que le calcul de cette correction du second ordre utilise les valeurs $c(x^{(k)} + s_x^{(k)})$ qu'il est donc nécessaire d'évaluer.

Certaines méthodes SQP avec recherche linéaire utilisent plutôt la correction $s_c^{(k)}$ pour effectuer une recherche unidimensionnelle sur la fonction de mérite $\Psi(x^{(k)} + \xi s_x^{(k)} + \xi^2 s_c^{(k)})$. Le terme « recherche linéaire » est cependant impropre, puisque la trajectoire décrite dans l'espace de conception n'est plus une droite mais une parabole. L'utilisation de cette trajectoire parabolique en lieu et place de la trajectoire rectiligne $x^{(k)} + \xi s_x^{(k)}$ autorise de plus grands pas de progression que la recherche linéaire « classique » pour une valeur de σ inchangée.

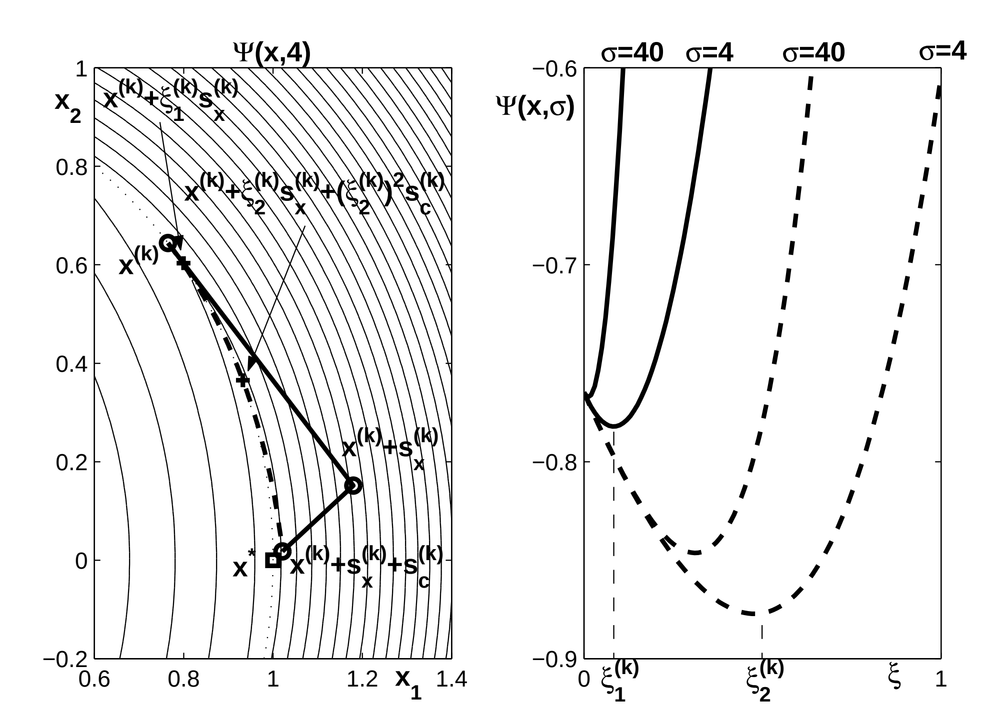


FIG. 8.5 – Illustration de la correction du second ordre pour le problème (8.19). La figure de gauche présente les contours de la fonction de mérite $\Psi(x)$ et la ligne discontinue en trait épais est la trajectoire parabolique $x^{(k)} + \xi s_x^{(k)} + \xi^2 s_c^{(k)}$ avec la variable $\xi \in [0,1]$. La figure de droite représente la valeur de la fonction de mérite le long de la trajectoire rectiligne $x^{(k)} + \xi s_x^{(k)}$ (trait continu), la valeur de la même fonction de mérite le long de la trajectoire parabolique $x^{(k)} + \xi s_x^{(k)} + \xi^2 s_c^{(k)}$ (trait discontinu) et leur minimum respectif $\xi_1^{(k)}$ et $\xi_2^{(k)}$. Nous pouvons constater que l'utilisation de la trajectoire parabolique autorise un pas de progression $\xi_2^{(k)}$ plus grand même pour des valeurs de σ plus importantes.

Exemple 8.5 *La figure 8.5 présente la correction du second ordre et la trajectoire parabolique correspondante pour le problème* (8.19) *autour du point* $x^{(k)} = (\cos\theta, \sin\theta)$ *pour lequel nous avions constaté un effet Maratos avec le pas* $s_x^{(k)}$*, solution du problème* (8.33)*. Sur cet exemple, la décomposition QR de la matrice* $G^{(k)}$ *est*

$$\begin{pmatrix} 2\cos\theta \\ 2\sin\theta \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \tag{8.51}$$

la correction du second ordre

$$s_c^{(k)} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} -\frac{1}{2}\sin^2\theta \\ 0 \end{pmatrix} = -\frac{1}{2}\sin^2\theta \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \tag{8.52}$$

et l'expression de la fonction unidimensionnelle le long de la trajectoire parabolique correspondante

$$\begin{aligned} \phi(\xi,\sigma) &= \Psi(x^{(k)} + \xi s_x^{(k)} + \xi^2 s_c^{(k)}) \\ &= (2+\sigma)\sin^4\theta \frac{\xi^4}{4} + \sin^2\theta\cos^2\theta \frac{\xi^2}{2} - \sin^2\theta\,\xi - \cos\theta. \end{aligned} \quad (8.53)$$

8.4 Conclusion

Nous avons pu voir, dans ce chapitre, que les méthodes de type SQP, outre leur simplicité, présentent des propriétés de convergence remarquables malgré le fait qu'il faut habituellement « mélanger » le Hessien de la fonction objectif et la courbure des contraintes dans la formulation du sous-problème quadratique. Nous avons également développé et analysé les propriétés d'une fonction de mérite de type ℓ_1 dans ce type d'algorithmes et son articulation avec des régions de confiance. Enfin, nous avons exposé les inconvénients de l'effet Maratos ainsi que la correction du second ordre traditionnellement mise en oeuvre pour l'éviter.

Nous nous sommes donc concentrés dans ce chapitre sur l'utilisation répandue d'approximations locales du second ordre pour la fonction objectif et du premier ordre pour les contraintes. Pour quelle raison le degré d'approximation est-il différent ? Pourquoi ne pas utiliser des approximations du second ordre également pour les contraintes ? La suite de notre travail porte précisément sur cette question tout en s'inspirant des techniques utilisée dans les cadre des méthodes SQP pour s'approcher, autant que faire se peut, des remarquables propriétés de convergence de ces algorithmes.

Chapitre 9

Description de l'algorithme UVQCQP

Nous avons envisagé, dans le chapitre précédent, l'utilisation d'approximations quadratiques pour la fonction objectif et pour les contraintes. Cette perspective prometteuse se heurte aux problèmes liés à la résolution du sous-problème local qui est un problème dit QCQP (*quadratically constrained quadratic programming*). Le chapitre qui suit a pour objet d'établir un algorithme de résolution de ce type de problèmes, permettant ainsi l'élaboration d'une méthode de type QCQP récursif ou *sequential* QCQP.

La technique de globalisation choisie est celle des régions de confiance. Vu la complexité potentielle du problème, nous avons choisi de caractériser la région de confiance par la norme dont la forme est la plus simple à prendre en compte, soit la norme ℓ_∞

$$\|s\|_k = \|s\|_\infty = \max_{i=1}^{n} \left|[s]_i\right| \tag{9.1}$$

qui donne à la région de confiance une forme d'hyper-boîte dont les faces sont parallèles aux axes[1]. La contrainte de confinement $\|s\|_k \leq \Delta^{(k)}$ se transforme en contraintes de bornes dans le problème local, ce qui lui confère l'indéniable avantage d'être aisément combinée avec de véritables contraintes de bornes.

Afin d'éviter les problèmes d'incompatibilité entre les contraintes de confinement et les contraintes réelles du problème, nous approchons le problème contraint par le biais de la fonction de pénalité ℓ_1. Celle-ci présente l'avantage d'être une fonction de pénalité exacte mais le désavantage de ne pas être en tout point différentiable. Afin de simplifier le problème, nous ne considérons que des contraintes

[1] Tout au long de ce chapitre les composantes d'un vecteur a ou d'une matrice A seront désignées par $[a]_i$ et $[A]_{i,j}$.

d'*inégalité convexes* écrites sous la forme générique

$$\phi_i(x) \leq 0, \qquad i = 1, 2, \ldots, m. \tag{9.2}$$

Cette hypothèse est bien entendu très restrictive mais nous aborderons le cas général dans le chapitre suivant.

L'algorithme développé dans ce chapitre pour résoudre le problème QCQP s'inspire des développements théoriques de Lemaréchal *et al.* [65] qui ont introduit une décomposition de $\mathbb{R}^n$ en deux sous-espaces : $\mathcal{U}$ dans lequel la fonction objectif est différentiable et $\mathcal{V}$ dans lequel elle ne l'est pas. C'est pour cette raison que l'algorithme porte le nom « UVQCQP ».

9.1 Trois sous-espaces orthogonaux.

La construction d'une fonction de mérite avec une pénalité de type ℓ_1 nous amène à considérer un problème d'optimisation de la forme[2]

$$\text{minimiser} \qquad \phi(x) = \phi_0(x) + \sum_{i=1}^{m} \max\left[0, \phi_i(x)\right] \tag{9.3}$$

$$\text{s.c.} \qquad x_L \leq x \leq x_U. \tag{9.4}$$

Les vecteurs x_L et $x_U \in \mathbb{R}^n$ sont des contraintes de bornes. Les fonctions $\phi_i(x)$ sont quadratiques et convexes, *i.e.*

$$\phi_i(x) = \frac{1}{2} x^T A_i x + x^T a_i + \alpha_i \tag{9.5}$$

avec $\alpha_i \in \mathbb{R}$, $a_i \in \mathbb{R}^n$ et $A_i \in \mathbb{R}^n_n$ pour $i = 0, \ldots, m$; les matrices A_i sont semi-définies positives.

Dans ce chapitre, l'*arrête i* désigne l'ensemble des points x tel que $\phi_i(x) = 0$. La fonction $\phi(x)$ est différentiable en dehors de ses arrêtes ; elle est continue mais non-différentiable sur ses arrêtes.

9.1.1 Le sous-espace $\mathcal{W}^{(k)}$.

La méthode développée est itérative. La présence de contraintes de bornes suggère l'utilisation d'une stratégie de contraintes actives ou de projection. Si, pour un i donné, $[x_L]_i = [x_U]_i$, la variable $[x]_i$ est fixée à cette valeur déterminée. Dans la suite du chapitre nous considérerons que $[x_L]_i < [x_U]_i$ pour $i = 1, \ldots, n$.

[2] Dans un souci de simplicité, le paramètre de pénalité apparaissant dans l'expression générale (8.15) est posé égal à l'unité.

La stratégie exacte d'activation et de désactivation des contraintes de bornes sera évoquée en détail ultérieurement. À ce stade, nous devons seulement tenir compte du fait qu'à une itération donnée k, certaines variables $[x]_i$ sont fixées à leur valeur minimum $[x_L]_i$, d'autres à leur valeur maximum $[x_U]_i$ et les dernières sont libres.

Nous définirons l'ensemble $\mathcal{W}^{(k)}$ comme le sous-espace de $\mathbb{R}^n$ généré par les vecteurs canoniques correspondant aux variables fixées. Une matrice comprenant une base orthonormée $W^{(k)}$ de ce sous-espace peut aisément être construite en utilisant les vecteurs canoniques

$$e_i = \begin{pmatrix} 0 & 0 & \dots & 1 & \dots & 0 \end{pmatrix}^T \tag{9.6}$$

pour chaque variable $[x]_i$ fixée à une de ses bornes. La dimension du complément orthogonal $\mathcal{W}^{(k)\perp}$ est la *dimension de travail*

$$\bar{n}^{(k)} = n - \dim \mathcal{W}^{(k)}. \tag{9.7}$$

Un élément donné x de $\mathbb{R}^n$ est aisément décomposé en $x = \tilde{x} + \bar{x}$ où $\tilde{x} \in \mathcal{W}^{(k)}$ et $\bar{x} \in \mathcal{W}^{(k)\perp}$.

Dans un souci de clarté de l'exposé et sans perte de généralité, nous supposerons que les variables fixées sont les premières et que les variables libres sont les suivantes. En conséquence, tout vecteur $x \in \mathbb{R}^n$ est séparable par bloc

$$x = \tilde{x} + \bar{x} = \begin{pmatrix} \tilde{x} \\ \bar{x} \end{pmatrix}. \tag{9.8}$$

et toute matrice de n lignes peut être partitionnée de façon semblable

$$G = \begin{pmatrix} \tilde{G} \\ \bar{G} \end{pmatrix}. \tag{9.9}$$

La matrice $W^{(k)}$ de dimension $n \times \dim \mathcal{W}^{(k)}$ constituée d'une base orthonormée de $\mathcal{W}^{(k)}$ s'écrit, quant à elle,

$$W^{(k)} = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} \\ 0 \end{pmatrix}. \tag{9.10}$$

9.1.2 Les sous-espaces $\mathcal{U}^{(k)}$ et $\mathcal{V}^{(k)}$.

Introduisons tout d'abord le concept développé par Lemaréchal *et al.* [65], également utilisé par Mifflin and Sagastizábal [73]. À un point $x \in \mathbb{R}^n$ donné et une fonction non-différentiable $\phi(x)$, on peut associer deux sous-espaces orthogonaux $\mathcal{U}(x)$ et $\mathcal{V}(x)$. Le sous-espace $\mathcal{V}(x)$ est défini comme l'enveloppe linéaire

$$\mathcal{V}(x) = \operatorname{lin}\{\partial\phi(x) - g\} \tag{9.11}$$

où $g \in \partial\phi(x)$ est un sous-gradient arbitraire. Le sous-espace $\mathcal{U}(x)$ est le complément orthogonal de $\mathcal{V}(x)$. Ces deux sous-ensembles sont définis de sorte que le sous-espace $\mathcal{U}(x)$ soit l'espace affin de la plus grande dimension dans lequel la fonction devient différentiable en x. Les noms $\mathcal{U}$ et $\mathcal{V}$ des deux ensembles résultent de cette propriété : le graphisme de la lettre $\mathcal{U}$ évoque une vallée différentiable tandis que celui de la lettre $\mathcal{V}$ présente un angle.

Si la décomposition $\mathcal{U}\mathcal{V}$ est appliquée pour l'analyse de la fonction $\phi(x)$ réduite au sous-espace des variables libres $\mathcal{W}^{(k)\perp}$, nous obtenons deux sous-espaces orthogonaux $\mathcal{U}^{(k)}$ et $\mathcal{V}^{(k)}$ qui sont eux-mêmes orthogonaux au sous-espace $\mathcal{W}^{(k)}$.

Le sous-espace $\mathcal{V}^{(k)}$ est donc défini comme

$$\mathcal{V}^{(k)} = \operatorname{lin}\left\{\partial\phi(x^{(k)}) - g^{(k)}\right\} \cap \mathcal{W}^{(k)\perp} \tag{9.12}$$

où $g^{(k)} \in \partial\phi(x^{(k)})$ est un sous-gradient arbitraire. Le sous-espace $\mathcal{U}^{(k)}$ est simplement le complément orthogonal de $\mathcal{V}^{(k)}$ dans $\mathcal{W}^{(k)\perp}$, *i.e.* tel que $\mathcal{U}^{(k)} \oplus \mathcal{V}^{(k)} = \mathcal{W}^{(k)\perp}$.

En tenant compte de la forme particulière (9.3) et (9.5) de $\phi(x)$, un sous-différentiel peut être facilement caractérisé. Le sous-différentiel $\partial\phi(x^{(k)})$ est l'ensemble de tous les sous-gradients en un point donné $x^{(k)}$. Le sous-différentiel d'une somme de fonctions convexes est la somme cartésienne de ses sous-différentiels. Le sous-gradient des fonctions quadratiques par morceaux $\max(0, \phi_i(x))$ au point $x^{(k)}$ s'écrit

$$\partial \max[0, \phi_i(x^{(k)})] = \begin{cases} \{0\} & \text{si } \phi(x^{(k)}) < 0, \\ \{\lambda \nabla_x \phi_i(x^{(k)}) : \lambda \in [0,1]\} & \text{si } \phi(x^{(k)}) = 0, \\ \{\nabla_x \phi_i(x^{(k)})\} & \text{si } \phi(x^{(k)}) > 0. \end{cases} \tag{9.13}$$

Si nous définissons les trois ensembles d'indices

$$\mathcal{N}^{(k)} = \{j : \phi_j(x^{(k)}) < 0, j = 1, \dots, m\}, \tag{9.14}$$

$$\mathcal{Z}^{(k)} = \{j : \phi_j(x^{(k)}) = 0, j = 1, \dots, m\}, \tag{9.15}$$

$$\mathcal{P}^{(k)} = \{j : \phi_j(x^{(k)}) > 0, j = 1, \dots, m\}, \tag{9.16}$$

le sous-différentiel de $\phi(x)$ au point $x^{(k)}$ peut s'écrire

$$\partial\phi(x^{(k)}) = \left\{ \nabla_x \phi_0(x^{(k)}) + \sum_{i \in \mathcal{P}^{(k)}} \nabla_x \phi_i(x^{(k)}) + \sum_{i \in \mathcal{Z}^{(k)}} \lambda_i \nabla_x \phi_i(x^{(k)}) : \lambda_i \in [0,1] \right\}. \tag{9.17}$$

Le vecteur

$$g^{(k)} = \nabla_x \phi_0(x^{(k)}) + \sum_{i \in \mathcal{P}^{(k)}} \nabla_x \phi_i(x^{(k)}) \tag{9.18}$$

est un sous-gradient particulier de $\partial\phi(x^{(k)})$. Conformément à sa définition (9.12), le sous-espace $\mathcal{V}^{(k)}$ s'écrit donc

$$\mathcal{V}^{(k)} = \text{lin}\left\{\sum_{i\in\mathcal{Z}^{(k)}} \lambda_i \nabla_x \phi_i(x^{(k)}) : \lambda_i \in [0,1]\right\} \cap \mathcal{W}^{(k)\perp}. \tag{9.19}$$

L'enveloppe linéaire d'un espace formulé de la sorte est très simple à obtenir : il suffit d'autoriser toutes les valeurs réelles pour les coefficients λ_i. En tenant compte de (9.5), nous obtenons

$$\mathcal{V}^{(k)} = \left\{\sum_{i\in\mathcal{Z}^{(k)}} \lambda_i g_i^{(k)} : \lambda_i \in \mathbb{R}\right\} \cap \mathcal{W}^{(k)\perp}. \tag{9.20}$$

avec

$$g_i^{(k)} = A_i x^{(k)} + a_i. \tag{9.21}$$

Dans les sous-espaces $\mathcal{W}^{(k)}$, $\mathcal{V}^{(k)}$ et $\mathcal{U}^{(k)}$, trois matrices $W^{(k)}$, $V^{(k)}$ et $U^{(k)}$ constituées de vecteurs de base orthonormés peuvent être construites. Ces matrices, respectivement de dimension $n\times\dim\mathcal{W}^{(k)}$, $n\times\dim\mathcal{V}^{(k)}$ et $n\times\dim\mathcal{U}^{(k)}$, sont obtenues grâce à un algorithme d'orthonormation en utilisant la matrice $G^{(k)}$ de dimension $n\times|\mathcal{Z}^{(k)}|$ formée par les vecteurs $g_i^{(k)}$ pour lesquels $i\in\mathcal{Z}^{(k)}$, *i.e.*

$$G^{(k)} = \left(g_i^{(k)}\right)_{i\in\mathcal{Z}^{(k)}}. \tag{9.22}$$

Pour ce faire, il convient d'opérer une factorisation QR sur la matrice composée par blocs $(W^{(k)}\ G^{(k)})$ de sorte que

$$\left(W^{(k)}\ G^{(k)}\right) = Q^{(k)} R^{(k)} P^{(k)} \tag{9.23}$$

où $Q^{(k)}$ est une matrice carrée orthogonale de dimension $n\times n$, $R^{(k)}$ une matrice triangulaire supérieure de dimension $n\times(\dim\mathcal{W}^{(k)}+|\mathcal{Z}^{(k)}|)$ et $P^{(k)}$ une matrice de permutation carrée de dimension $\dim\mathcal{W}^{(k)}+|\mathcal{Z}^{(k)}|$. Cette expression peut être décomposée par blocs

$$\left(W^{(k)}\ G^{(k)}\right) = \begin{pmatrix} I_{\dim\mathcal{W}^{(k)}} & \tilde{G}^{(k)} \\ 0 & \bar{G}^{(k)} \end{pmatrix} \tag{9.24}$$

où $\tilde{G}^{(k)}$ et $\bar{G}^{(k)}$ sont des matrices de $|\mathcal{Z}^{(k)}|$ colonnes. Les éléments de la décomposition QR s'écrivent, quant à eux,

$$Q^{(k)} = \begin{pmatrix} I_{\dim\mathcal{W}^{(k)}} & 0 \\ 0 & \bar{Q}^{(k)} \end{pmatrix}, \tag{9.25}$$

$$R^{(k)} = \begin{pmatrix} I_{\dim\mathcal{W}^{(k)}} & \tilde{G}^{(k)}\bar{P}^{(k)} \\ 0 & \bar{R}^{(k)} \end{pmatrix}, \tag{9.26}$$

$$P^{(k)} = \begin{pmatrix} I_{\dim\mathcal{W}^{(k)}} & 0 \\ 0 & \bar{P}^{(k)} \end{pmatrix}. \tag{9.27}$$

La décomposition QR complète peut donc être exprimée avec les éléments de décomposition QR de $\bar{G}^{(k)}$ puisque $\bar{G}^{(k)} = \bar{Q}^{(k)}\bar{R}^{(k)}\bar{P}^{(k)}$. Notons que $\bar{P}^{(k)}$ est une matrice de permutation carrée de dimension $|\mathcal{Z}^{(k)}|$ telle que la condition de décroissance diagonale

$$\left|[\bar{R}^{(k)}]_{1,1}\right| \geq \left|[\bar{R}^{(k)}]_{2,2}\right| \geq \ldots \geq \left|[\bar{R}^{(k)}]_{\ell^{(k)},\ell^{(k)}}\right| \tag{9.28}$$

soit satisfaite. L'entier positif $\ell^{(k)}$ est défini comme le plus petit des deux entiers positifs $|\mathcal{Z}^{(k)}|$ et $\bar{n}^{(k)}$. Dans la suite de ce chapitre, nous supposerons qu'aucune permutation n'est nécessaire. Ceci peut se faire sans perte de généralité puisque l'ordre des colonnes constituant la matrice (9.22) n'a pas été imposé. La matrice de permutation est donc égale à la matrice identité $\bar{P}^{(k)} = I_{|\mathcal{Z}^{(k)}|}$ et la décomposition QR s'écrit

$$Q^{(k)} = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & 0 \\ 0 & \bar{Q}^{(k)} \end{pmatrix}, \tag{9.29}$$

$$R^{(k)} = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & \tilde{G}^{(k)} \\ 0 & \bar{R}^{(k)} \end{pmatrix}. \tag{9.30}$$

La dimension du sous-espace $\mathcal{V}^{(k)}$ correspond au rang de la matrice $\bar{R}^{(k)}$, *i.e.* le plus grand entier i tel que[3]

$$\left|[\bar{R}^{(k)}]_{i,i}\right| > 0. \tag{9.31}$$

La matrice $Q^{(k)}$ peut dès lors être découpée en trois matrices $W^{(k)}$, $V^{(k)}$ et $U^{(k)}$,

$$Q^{(k)} = \left(W^{(k)}\ V^{(k)}\ U^{(k)}\right) = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & 0 & 0 \\ 0 & \bar{V}^{(k)} & \bar{U}^{(k)} \end{pmatrix} \tag{9.32}$$

avec

$$\bar{Q}^{(k)} = \left(\bar{V}^{(k)}\ \bar{U}^{(k)}\right). \tag{9.33}$$

La propriété d'orthogonalité de $Q^{(k)}$ mène aisément à l'expression suivante

$$\begin{pmatrix} W^{(k)T}W^{(k)} & W^{(k)T}V^{(k)} & W^{(k)T}U^{(k)} \\ V^{(k)T}W^{(k)} & V^{(k)T}V^{(k)} & V^{(k)T}U^{(k)} \\ U^{(k)T}W^{(k)} & U^{(k)T}V^{(k)} & U^{(k)T}U^{(k)} \end{pmatrix} = I_n \tag{9.34}$$

ou

$$\begin{pmatrix} \bar{V}^{(k)T}\bar{V}^{(k)} & \bar{V}^{(k)T}\bar{U}^{(k)} \\ \bar{U}^{(k)T}\bar{V}^{(k)} & \bar{U}^{(k)T}\bar{U}^{(k)} \end{pmatrix} = \begin{pmatrix} I_{\dim \mathcal{V}^{(k)}} & 0 \\ 0 & I_{\dim \mathcal{U}^{(k)}} \end{pmatrix}. \tag{9.35}$$

[3]Naturellement, les implémentations pratiques utilisent en réalité un petit seuil positif ε en lieu et place de zéro.

L'orthogonalité de $\mathcal{U}^{(k)}$ par rapport à $\mathcal{V}^{(k)}$ conjuguée à sa définition (9.20) nous donne, quant à elle, l'expression

$$g_i^{(k)T} U^{(k)} = 0 \qquad \text{pour } i \in \mathcal{Z}^{(k)} \tag{9.36}$$

ou

$$\bar{g}_i^{(k)T} \bar{U}^{(k)} = 0 \qquad \text{pour } i \in \mathcal{Z}^{(k)}. \tag{9.37}$$

Pour rappel, les vecteurs $\bar{g}_i^{(k)}$ sont le résultat de la décomposition par blocs (9.8) de $g_i^{(k)}$.

Exemple 9.1 *Afin d'illustrer les espaces $\mathcal{U}^{(k)}$, $\mathcal{V}^{(k)}$ et $\mathcal{W}^{(k)}$, envisageons la fonction $\phi(x)$ de trois variables*

$$\phi(x) = \underbrace{-\frac{x_1}{2} - x_2 - x_3}_{\phi_0(x)} + \max[0, \underbrace{x_2^2 + x_3^2 - 1}_{\phi_1(x)}] + \max[0, \underbrace{x_2^2 + x_3^2 + 2x_3 - 3}_{\phi_2(x)}] \tag{9.38}$$

au point $x^{(k)} = (0,0,1)$. Supposons que seule la contrainte de borne inférieure pour la troisième variable x_1 soit active, i.e. $[x_L]_1 = 0$.

Une seule contrainte de borne étant active, la dimension de l'espace $\mathcal{W}^{(k)}$ est égale à l'unité et la dimension de travail (9.7) *est égale à deux. La matrice $W^{(k)}$ est donc le vecteur colonne $(1\ 0\ 0)^T$.*

Vérifiant que $\phi_1(x^{(k)}) = 0$ et $\phi_2(x^{(k)}) = 0$, nous obtenons

$$\mathcal{Z}^{(k)} = \{1,2\} \tag{9.39}$$

tandis que les ensembles $\mathcal{N}^{(k)}$ et $\mathcal{P}^{(k)}$ sont vides. Nous pouvons alors construire la matrice $G^{(k)}$ de dimension $n \times |\mathcal{Z}^{(k)}|$ constituée des gradients (9.21)

$$G^{(k)} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 2 & 4 \end{pmatrix} \tag{9.40}$$

dont on extrait la matrice de dimension $\bar{n}^{(k)} \times |\mathcal{Z}^{(k)}|$

$$\bar{G}^{(k)} = \begin{pmatrix} 0 & 0 \\ 2 & 4 \end{pmatrix}. \tag{9.41}$$

La décomposition QR de $\bar{G}^{(k)}$ nous donne

$$\bar{G}^{(k)} = \bar{Q}^{(k)} \bar{R}^{(k)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 2 & 4 \\ 0 & 0 \end{pmatrix}. \tag{9.42}$$

La matrice $R^{(k)}$ est de rang un, l'espace $\mathcal{V}^{(k)}$ est donc également de dimension un. La première colonne de $\overline{Q}^{(k)}$ constitue la matrice $\overline{V}^{(k)}$ alors que la deuxième est la matrice $\overline{U}^{(k)}$. Les matrices $V^{(k)}$ et $U^{(k)}$ correspondantes sont respectivement $(0\ 0\ 1)^T$ et $(0\ 1\ 0)^T$.

La fonction $\phi(x)$ est représentée sur le premier graphique de la figure 9.1 pour une valeur constante $x_1 = 0$. Pour cet exemple, les espaces $\mathcal{U}(x^{(k)})$ et $\mathcal{V}(x^{(k)})$ correspondants sont représentés par des lignes parallèles aux axes tandis que $\mathcal{W}^{(k)}$ devrait être une ligne droite perpendiculaire au plan de la feuille et passant par $x^{(k)}$. Les valeurs de la fonction $\phi(x)$ dans ces sous-espaces peuvent être représentées (voir figure 9.1) en fonction du paramètre ξ

$$x(\xi) = x^{(k)} + \xi e_i \tag{9.43}$$

où e_i est $e_{\mathcal{U}}$, $e_{\mathcal{V}}$ ou $e_{\mathcal{W}}$, des vecteurs unitaires dans les espaces $\mathcal{U}(x^{(k)})$, $\mathcal{V}(x^{(k)})$ et $\mathcal{W}(x^{(k)})$. Nous constatons que les tracés des deux premières fonctions présentent, au voisinage du point $x^{(k)}$, un graphisme se rapprochant respectivement de celui des lettres $\mathcal{U}$ et $\mathcal{V}$. Remarquons que la fonction ainsi obtenue n'est pas partout différentiable dans l'espace $\mathcal{V}$ alors qu'elle l'est dans l'espace $\mathcal{U}$.

9.2 Directions de descente.

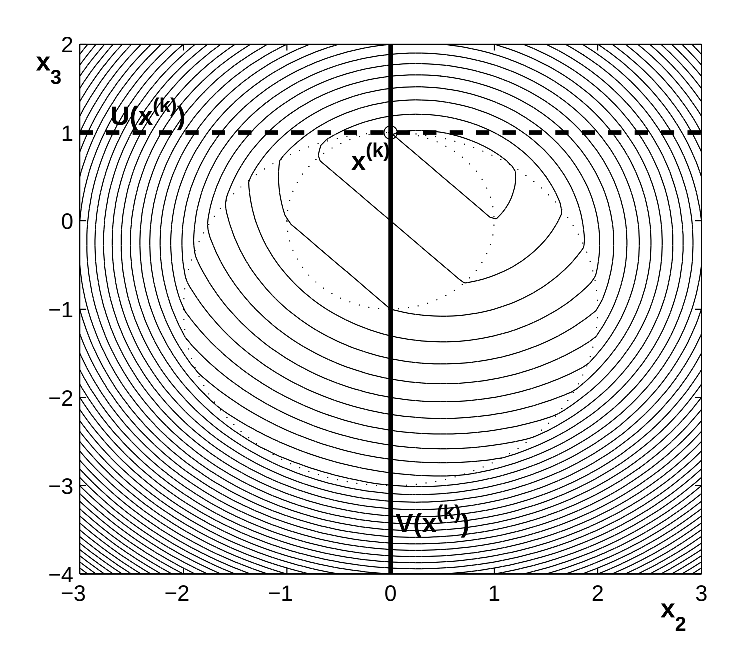
L'algorithme proposé est basé sur l'obtention de différentes directions de descente. Ces directions se calculent sur base d'une fonction approchée $\varphi^{(k)}(x)$ qui est parfaitement égale à $\phi(x)$ dans un voisinage (suffisamment petit) de l'itéré $x^{(k)}$

$$\varphi^{(k)}(x) = \phi_0(x) + \sum_{i\in\mathcal{P}^{(k)}} \phi_i(x) + \sum_{i\in\mathcal{Z}^{(k)}} \max\left[0, \phi_i(x)\right]. \tag{9.44}$$

Cette fonction approchée est construite pour éliminer un maximum des discontinuités possibles dans les dérivées. Les termes de la fonction (9.3) sont remplacés par :

- la fonction identiquement nulle si $\phi_j(x^{(k)}) < 0$, c'est-à-dire si $j \in \mathcal{N}^{(k)}$ (la contrainte correspondante (9.2) est satisfaite) ;
- par $\phi_j(x)$ si $\phi_j(x^{(k)}) > 0$, c'est-à-dire si $j \in \mathcal{P}^{(k)}$ (la contrainte correspondante (9.2) est violée) ; ou
- par $\max\left[0, \phi_i(x)\right]$ si $\phi_j(x^{(k)}) = 0$, c'est-à-dire si $j \in \mathcal{Z}^{(k)}$ (la contrainte correspondante (9.2) est active).

Nous ne conservons donc intacts que les termes provoquant effectivement une « cassure de pente » de $\phi(x)$ en $x^{(k)}$, les autres sont remplacés par un terme continûment dérivable.



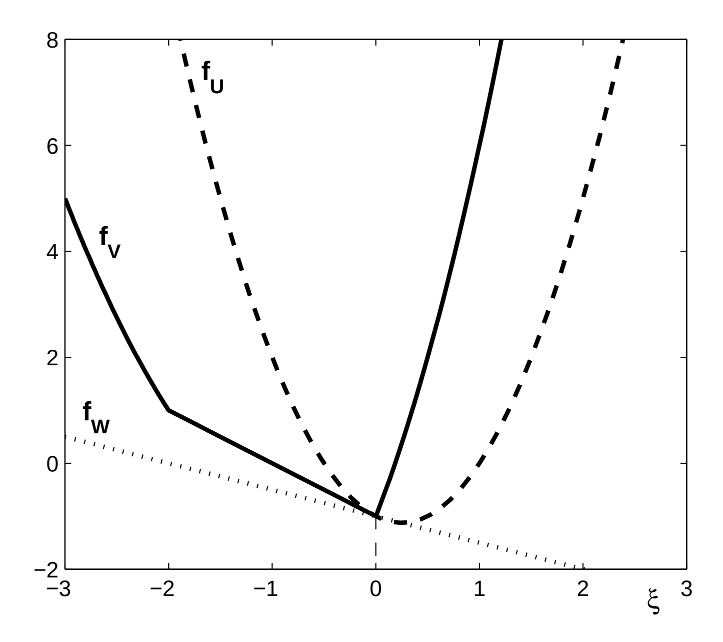


FIG. 9.1 – Figure du haut : illustration des espaces $\mathcal{U}$ (trait épais discontinu) et $\mathcal{V}$ (trait épais continu) au point $x^{(1)} = (0,0,1)$ pour la fonction (9.38) dans le plan $x_1 = 0$. Figure du bas : représentations de la fonction le long des espaces $\mathcal{U}$ (trait discontinu), $\mathcal{V}$ (trait discontinu) et $\mathcal{W}$ (trait pointillé).

Exemple 9.2 *Afin d'illustrer le concept de fonction approchée* $\varphi^{(k)}$*, envisageons la fonction* $\phi(x)$ *de trois variables* (9.38) *représentée sur le premier graphique de la figure 9.1 dans le plan* $x_1 = 0$*. La fonction approchée* $\varphi^{(1)}(x)$ *correspondant au point* $x^{(1)} = (0,0,1)$ *est égale à* $\phi(x)$ *en tout point de* $\mathbb{R}^3$*. En effet,* $\phi_1(x^{(1)}) = 0$ *et* $\phi_2(x^{(1)}) = 0$ *et l'ensemble* $\mathcal{Z}^{(1)} = \{1,2\}$ *et les ensembles* $\mathcal{P}^{(1)}$ *et* $\mathcal{N}^{(1)}$ *sont vides. Pour d'autres points, la situation est différente.*

Au point $x^{(2)} = (0,0,2)$ *pour lequel les deux fonctions* ϕ_1 *et* ϕ_2 *sont strictement négatives, l'application de la définition* (9.44) *permet de conclure que la fonction approchée* $\varphi^{(2)}(x)$ *est égale à* $\phi_0(x)$*, comme l'illustre le deuxième graphique de la figure 9.2. Remarquons que* $\phi(x)$ *et* $\varphi^{(2)}(x)$ *sont bel et bien égales dans un voisinage de* $x^{(2)}$*, voisinage qui s'étend à l'ensemble des points pour lesquels* $\phi_1(x)$ *et* $\phi_2(x)$ *est strictement négatives.*

De la même manière, la fonction approchée $\varphi^{(3)}(x)$ *correspondant au point* $x^{(3)} = (0,0,-1)$ *s'écrit*

$$\varphi^{(3)}(x) = \phi_0(x) + \max[0, \phi_1(x)] \tag{9.45}$$

puisque $\phi_1(x^{(3)}) = 0$ *s'annule et* $\phi_2(x^{(3)}) < 0$*. La fonction* $\varphi^{(3)}(x)$ *ainsi que les fonctions* $\varphi^{(4)}(x)$*,* $\varphi^{(5)}(x)$ *et* $\varphi^{(6)}(x)$ *correspondant respectivement aux points*

$$x^{(4)} = (0,0,-2),\ x^{(5)} = (0,0,-3) \text{ et } x^{(6)} = (0,0,-4)$$

sont représentées à la figure 9.2.

Chaque fonction quadratique $\phi_i(x)$ peut être écrite autour de $x^{(k)}$

$$\phi_i(x) = \phi_i(x^{(k)}) + s^T g_i^{(k)} + \frac{1}{2} s^T A_i s \tag{9.46}$$

où $s = x - x^{(k)}$. Notons que $\phi_i(x^{(k)}) = 0$ quand $i \in \mathcal{Z}^{(k)}$, par définition de l'ensemble $\mathcal{Z}^{(k)}$. Notre but est de trouver une direction de descente $s^{(k)}$ pour $\phi(x)$. Nous prouvons aisément que $s^{(k)}$ est une direction de descente pour $\phi(x)$ si et seulement si elle l'est également pour $\varphi^{(k)}(x)$. Ce problème sera envisagé dans chacun des sous-espaces $\mathcal{U}^{(k)}$, $\mathcal{V}^{(k)}$ et $\mathcal{W}^{(k)}$.

9.2.1 La direction de descente en $\mathcal{U}^{(k)}$.

Si nous restreignons le choix de la direction de descente dans le sous-espace $\mathcal{U}^{(k)}$, le vecteur s doit nécessairement être de la forme $s = U^{(k)}u$ où $u \in \mathbb{R}^{\dim \mathcal{U}^{(k)}}$. En tenant compte des expressions (9.36) et (9.46), la restriction dans $\mathcal{U}^{(k)}$ de la

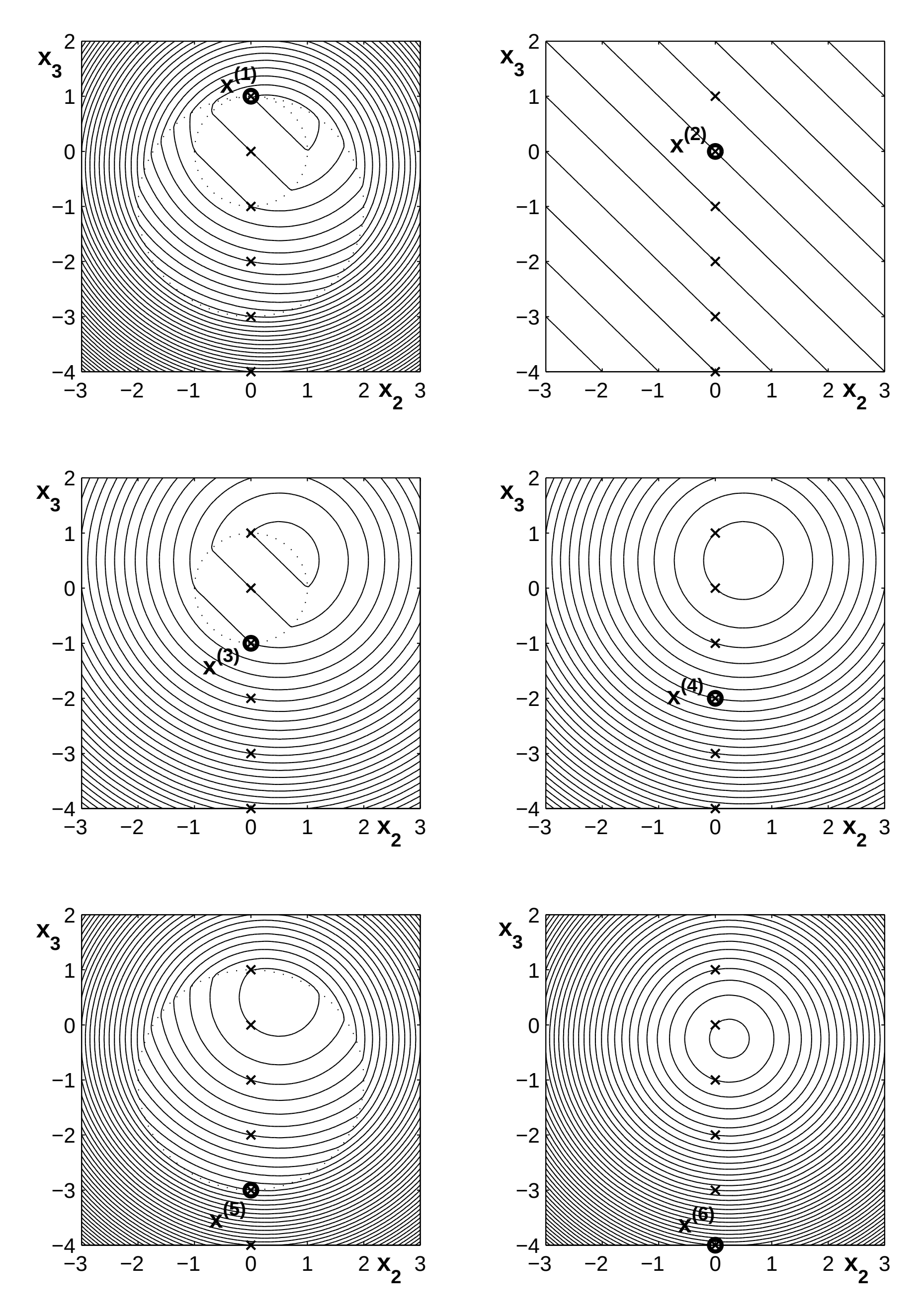


FIG. 9.2 – Représentation, dans le plan $x_1 = 0$, de la fonction approchée $\varphi^{(k)}(x)$ de (9.38) pour les six points marqués d'une croix. Les lignes pointillées indiquent les arêtes. La croix encerclée désigne le point $x^{(k)}$ autour duquel la fonction $\varphi^{(k)}$ a été construite. Constatons que la première figure représente à la fois les fonction $\phi(x)$ et $\varphi^{(1)}(x)$ puisque ces deux fonctions sont égales en tout point.

fonction (9.44) peut être écrite sous la forme

$$\begin{aligned}
\varphi_{\mathcal{U}}^{(k)}(u) &= \varphi^{(k)}(x^{(k)}+U^{(k)}u) \\
&= \phi(x^{(k)})+g_0^{(k)T}U^{(k)}u+\frac{1}{2}u^T\,U^{(k)T}A_0U^{(k)}\,u \\
&\quad+\sum_{i\in\mathcal{P}^{(k)}} g_i^{(k)T}U^{(k)}u+\sum_{i\in\mathcal{P}^{(k)}}\frac{1}{2}u^TU^{(k)T}A_iU^{(k)}u \\
&\quad+\sum_{i\in\mathcal{Z}^{(k)}}\max\left(0,\frac{1}{2}u^T\,U^{(k)T}A_iU^{(k)}\,u\right). \qquad (9.47)
\end{aligned}$$

Notons que $u^TU^{(k)T}A_iU^{(k)}u \geq 0$ en raison de la semi-définie positivité de A_i. En conséquence, les fonctions maximum dans (9.47) sont toujours égales à leur second argument et $\varphi_{\mathcal{U}}^{(k)}(u)$ est une fonction quadratique qui peut s'écrire simplement

$$\varphi_{\mathcal{U}}^{(k)}(u)=\phi(x^{(k)})+u^Ta_{\mathcal{U}}^{(k)}+\frac{1}{2}u^TA_{\mathcal{U}}^{(k)}u \qquad (9.48)$$

où

$$a_{\mathcal{U}}^{(k)}=U^{(k)T}\left(g_0^{(k)}+\sum_{i\in\mathcal{P}^{(k)}}g_i^{(k)}\right)=U^{(k)T}g^{(k)} \qquad (9.49)$$

et

$$A_{\mathcal{U}}^{(k)}=U^{(k)T}\left(A_0+\sum_{i\in\mathcal{P}^{(k)}}A_i+\sum_{i\in\mathcal{Z}^{(k)}}A_i\right)U^{(k)}=U^{(k)T}A^{(k)}U^{(k)}. \qquad (9.50)$$

Nous définissons $s_{\mathcal{U}}^{(k)}$, *la direction de descente en* $\mathcal{U}^{(k)}$, comme le produit $U^{(k)}u^{(k)}$ avec

$$\begin{aligned}
u^{(k)} &\in \arg\min_u \varphi_{\mathcal{U}}^{(k)}(u) && (9.51)\\
\text{s. c.}\quad & U^{(k)}u \geq x_L - x^{(k)}, && (9.52)\\
& U^{(k)}u \leq x_U - x^{(k)}. && (9.53)
\end{aligned}$$

Les contraintes (9.52) et (9.53) constituent une transposition des contraintes de bornes (9.4) dans le sous-espace $\mathcal{U}^{(k)}$. Toutefois, les contraintes de bornes actives ne doivent pas entrer en ligne de compte puisque le sous-espace $\mathcal{U}^{(k)}$ est orthogonal à $\mathcal{W}^{(k)}$, les contraintes (9.52) et (9.53) peuvent donc être reformulées

$$\begin{aligned}
\bar{U}^{(k)}u &\geq \bar{x}_L-\bar{x}^{(k)}, && (9.54)\\
\bar{U}^{(k)}u &\leq \bar{x}_U-\bar{x}^{(k)}. && (9.55)
\end{aligned}$$

Le problème (9.51) est quadratique, convexe et linéairement contraint et la littérature scientifique regorge de méthodes efficaces pour le résoudre (*e.g.* voir Fletcher [31] ou Boland [4]).

Exemple 9.3 *Calculons la direction de descente en $\mathcal{U}^{(k)}$ du problème* (9.38) *auquel nous ajoutons les contraintes de bornes*

$$\begin{pmatrix} 0 \\ -3 \\ -2 \end{pmatrix} \leq x \leq \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix}. \tag{9.56}$$

L'itéré courant est $x^{(k)} = (0,0,1)$ et la contrainte de borne inférieure sur la première composante est active. Pour obtenir la direction de descente en $\mathcal{U}^{(k)}$, il suffit de construire

$$g^{(k)} = g_0 = \begin{pmatrix} -1/2 \\ -1 \\ -1 \end{pmatrix}, \tag{9.57}$$

$$a_{\mathcal{U}}^{(k)} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} -1/2 \\ -1 \\ -1 \end{pmatrix} = -1 \tag{9.58}$$

et

$$A^{(k)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix} \tag{9.59}$$

puis

$$A_{\mathcal{U}}^{(k)} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} = 4. \tag{9.60}$$

La direction de descente recherchée est dès lors

$$s_{\mathcal{U}}^{(k)} = \begin{pmatrix} 0 \\ 1/4 \\ 0 \end{pmatrix} \tag{9.61}$$

puisque $u^{(k)} = 1/4$ est l'argument minimum de la fonction

$$\varphi_{\mathcal{U}}^{(k)}(u) = -1 - u + 2u^2 \tag{9.62}$$

soumis aux contraintes

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} u \geq \begin{pmatrix} -3 \\ -2 \end{pmatrix} \qquad \textit{et} \qquad \begin{pmatrix} 1 \\ 0 \end{pmatrix} u \leq \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \tag{9.63}$$

i.e. $-3 \leq u \leq 3$.

9.2.2 La direction de descente en $\mathcal{V}^{(k)}$.

Une direction dans le sous-espace $\mathcal{V}^{(k)}$ s'écrit $s = V^{(k)}v$ avec $v \in \mathbb{R}^{\dim \mathcal{V}^{(k)}}$. Nous pouvons, de la même manière que dans le sous-espace $\mathcal{U}^{(k)}$, développer chaque fonction $\phi_i(x)$ sous la forme (9.46) de sorte que la fonction (9.44) devienne

$$\begin{aligned}
\varphi_{\mathcal{V}}^{(k)}(v) &= \varphi^{(k)}(x^{(k)} + V^{(k)}v) \\
&= \phi(x^{(k)}) + v^T a_{\mathcal{V}}^{(k)} + \frac{1}{2} v^T A_{\mathcal{V}}^{(k)} v \\
&\quad + \sum_{i \in \mathcal{Z}^{(k)}} \max\left(0, v^T V^{(k)T} g_i^{(k)} + \frac{1}{2} v^T V^{(k)T} A_i V^{(k)} v\right) \qquad (9.64)
\end{aligned}$$

où

$$a_{\mathcal{V}}^{(k)} = V^{(k)T}\left(g_0^{(k)} + \sum_{i \in \mathcal{P}^{(k)}} g_i^{(k)}\right) = V^{(k)T} g^{(k)}, \qquad (9.65)$$

$$A_{\mathcal{V}}^{(k)} = V^{(k)T}\left(A_0 + \sum_{i \in \mathcal{P}^{(k)}} A_i\right) V^{(k)}. \qquad (9.66)$$

La direction de plus grande pente $v^{(k)}$ pour cette fonction non-différentiable est obtenue en prenant l'opposé du sous-gradient

$$\partial \varphi_{\mathcal{V}}^{(k)}(0) = \left\{ a_{\mathcal{V}}^{(k)} + \sum_{i \in \mathcal{Z}^{(k)}} [l]_i V^{(k)T} g_i^{(k)} : [l]_i \in [0,1] \right\} \qquad (9.67)$$

de norme minimale (voir section 3.1). Construisons la matrice $G_{\mathcal{V}}^{(k)}$ de dimension $(\dim \mathcal{V}^{(k)} \times |\mathcal{Z}^{(k)}|)$ formée en prenant les vecteurs $V^{(k)T} g_i^{(k)}$, $i \in \mathcal{Z}^{(k)}$ comme colonnes

$$G_{\mathcal{V}}^{(k)} = \left(V^{(k)T} g_i^{(k)}\right)_{i \in \mathcal{Z}^{(k)}} = V^{(k)T} G^{(k)}. \qquad (9.68)$$

Chaque sous-gradient au point $v = 0$ peut s'écrire

$$a_{\mathcal{V}}^{(k)} + G_{\mathcal{V}}^{(k)} l \qquad (9.69)$$

où le vecteur $l \in \mathbb{R}^{|\mathcal{Z}^{(k)}|}$. Le sous-gradient de norme minimale est donc obtenu comme solution du problème d'optimisation

$$\begin{aligned}
l^{(k)} &\in \arg\min \left(l^T G_{\mathcal{V}}^{(k)T} a_{\mathcal{V}}^{(k)} + \frac{1}{2} l^T G_{\mathcal{V}}^{(k)T} G_{\mathcal{V}}^{(k)} l \right), \qquad (9.70) \\
\text{s. c.} \quad & 0 \leq l \leq 1. \qquad (9.71)
\end{aligned}$$

Ce problème est convexe, quadratique et soumis à des contraintes linéaires : il peut être résolu de la même manière que (9.51). La *direction de descente en* $\mathcal{V}^{(k)}$ est alors la direction dans $\mathbb{R}^n$ qui correspond à $v^{(k)}$, l'opposé du sous-gradient de norme minimum

$$s_{\mathcal{V}}^{(k)} = V^{(k)} v^{(k)} = -V^{(k)} \left(a_{\mathcal{V}}^{(k)} + G_{\mathcal{V}}^{(k)} l^{(k)} \right). \tag{9.72}$$

Exemple 9.4 *Calculons la direction de descente en* $\mathcal{V}^{(k)}$ *du problème* (9.38) *à l'itéré* $x^{(k)} = (0,0,1)$. *Comme précédemment, la contrainte de borne inférieure sur la première composante est active. Pour obtenir la direction de descente en* $\mathcal{V}^{(k)}$*, il suffit de construire*

$$a_{\mathcal{V}}^{(k)} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/2 \\ -1 \\ -1 \end{pmatrix} = -1 \tag{9.73}$$

et la matrice

$$G_{\mathcal{V}}^{(k)} = \left(\begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \; \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix} \right) = \begin{pmatrix} 2 & 4 \end{pmatrix}. \tag{9.74}$$

La norme à minimiser d'un sous-gradient quelconque s'écrit donc

$$l^T \begin{pmatrix} -2 \\ -4 \end{pmatrix} + \frac{1}{2} l^T \begin{pmatrix} 4 & 8 \\ 8 & 16 \end{pmatrix} l \tag{9.75}$$

sous les contraintes $0 \leq l \leq 1$. *Le minimum est notamment obtenu pour*

$$l^{(k)} = \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \tag{9.76}$$

et la direction de descente correspondante est

$$s_{\mathcal{V}}^{(k)} = -\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \left(-1 + \begin{pmatrix} 2 & 4 \end{pmatrix} \begin{pmatrix} 1/2 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{9.77}$$

Comme attendu, il n'y a donc pas de descente possible dans l'espace $\mathcal{V}^{(k)}$ *(voir figure 9.1).*

9.2.3 La direction de descente en $\mathcal{W}^{(k)}$.

De manière tout à fait similaire à la direction de descente en $\mathcal{V}^{(k)}$, une direction de descente dans le sous-espace $\mathcal{W}^{(k)}$ s'exprime sous la forme $s = W^{(k)}w$ où $w \in \mathbb{R}^{\dim \mathcal{W}^{(k)}}$. En exprimant à nouveau chaque fonction ϕ_i sous la forme (9.46), la fonction (9.44) devient

$$\begin{aligned} \varphi_{\mathcal{W}}^{(k)}(w) &= \varphi^{(k)}(x^{(k)} + W^{(k)}w) \\ &= \phi(x^{(k)}) + w^T a_{\mathcal{W}}^{(k)} + \frac{1}{2} w^T A_{\mathcal{W}}^{(k)} w \\ &\quad + \sum_{i \in \mathcal{Z}^{(k)}} \max\left(0, w^T W^{(k)T} g_i^{(k)} + \frac{1}{2} w^T W^{(k)T} A_i W^{(k)} w\right) \end{aligned} \tag{9.78}$$

où

$$a_{\mathcal{W}}^{(k)} = W^{(k)T}\left(g_0^{(k)} + \sum_{i \in \mathcal{P}^{(k)}} g_i^{(k)}\right) = W^{(k)T} g^{(k)}, \tag{9.79}$$

$$A_{\mathcal{W}}^{(k)} = W^{(k)T}\left(A_0 + \sum_{i \in \mathcal{P}^{(k)}} A_i\right) W^{(k)}. \tag{9.80}$$

La direction de plus grande pente $w^{(k)}$ pour cette fonction non-différentiable est obtenue en prenant l'opposé du sous-gradient dont la norme est minimale (voir section 3.1)

$$\partial \varphi_{\mathcal{W}}^{(k)}(0) = \left\{ a_{\mathcal{W}}^{(k)} + \sum_{i \in \mathcal{Z}^{(k)}} [t]_i W^{(k)T} g_i^{(k)} : [t]_i \in [0,1] \right\}. \tag{9.81}$$

Construisons la matrice $G_{\mathcal{W}}^{(k)}$ de dimension $(\dim \mathcal{W}^{(k)} \times |\mathcal{Z}^{(k)}|)$ formée par les vecteurs $W^{(k)T} g_i^{(k)}$, $i \in \mathcal{Z}^{(k)}$

$$G_{\mathcal{W}}^{(k)} = \left(W^{(k)T} g_i^{(k)}\right)_{i \in \mathcal{Z}^{(k)}} = W^{(k)T} G^{(k)}. \tag{9.82}$$

Chaque sous-gradient peut s'écrire $a_{\mathcal{W}}^{(k)} + G_{\mathcal{W}}^{(k)} t$ où $t \in \mathbb{R}^{|\mathcal{Z}^{(k)}|}$. Le gradient de norme minimum est obtenu en résolvant le problème d'optimisation

$$t^{(k)} \in \arg\min \left(t^T G_{\mathcal{W}}^{(k)T} a_{\mathcal{W}}^{(k)} + \frac{1}{2} t^T G_{\mathcal{W}}^{(k)T} G_{\mathcal{W}}^{(k)} t \right), \tag{9.83}$$

$$\text{s. c.} \quad 0 \le t \le 1. \tag{9.84}$$

C'est un problème convexe, quadratique et linéairement contraint. La direction de descente en $\mathcal{W}^{(k)}$ est l'opposé de la direction correspondant à ce sous-gradient

$$s_{\mathcal{W}}^{(k)} = W^{(k)} w^{(k)} = -W^{(k)} \left(a_{\mathcal{W}}^{(k)} + G_{\mathcal{W}}^{(k)} t^{(k)} \right). \tag{9.85}$$

Exemple 9.5 *Calculons la direction de descente en $\mathcal{W}^{(k)}$ pour le problème* (9.38) *à l'itéré $x^{(k)} = (0,0,1)$. Comme précédemment, la contrainte de borne inférieure sur la première composante est active. Pour obtenir la direction de descente en $\mathcal{W}^{(k)}$, il suffit de construire*

$$a_{\mathcal{W}}^{(k)} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} -1/2 \\ -1 \\ -1 \end{pmatrix} = -1/2 \tag{9.86}$$

et la matrice

$$G_{\mathcal{W}}^{(k)} = \left(\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \ \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix} \right) = \begin{pmatrix} 0 & 0 \end{pmatrix}. \tag{9.87}$$

La norme à minimiser d'un sous-gradient quelconque s'écrit donc

$$t^T \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{2} t^T \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} t \tag{9.88}$$

sous les contraintes $0 \leq t \leq 1$. Le minimum est notamment obtenu pour

$$t^{(k)} = (0\ 0)^T \tag{9.89}$$

et la direction de descente correspondante est

$$s_{\mathcal{W}}^{(k)} = -\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \left(-1/2 + \begin{pmatrix} 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right) = \begin{pmatrix} 1/2 \\ 0 \\ 0 \end{pmatrix}. \tag{9.90}$$

9.3 Description d'une itération de base.

L'algorithme se base sur une approche relativement simple. Pour un itéré $x^{(k)}$, les directions de descente en $\mathcal{U}^{(k)}$, $\mathcal{V}^{(k)}$ et $\mathcal{W}^{(k)}$ sont calculées et explorées si celles-ci s'avèrent non-nulles.

Nous avons déjà signalé dans la section 9.1.1 que les contraintes de bornes étaient prises en compte grâce à une stratégie de contraintes actives. L'initialisation de l'ensemble des contraintes actives est simplement effectuée en inspectant l'estimation de départ $x^{(0)}$. Si une des composantes de $x^{(0)}$ correspond à une de ses bornes x_L ou x_U, la contrainte est *activée*, ce qui signifie que cette composante est bloquée jusqu'à nouvel ordre.

Algorithme 9.1 *Le schéma simplifié d'une itération de l'algorithme est le suivant.*

Étape 1 : Identification des arêtes actives. *Construire les ensembles $\mathcal{N}^{(k)}$, $\mathcal{Z}^{(k)}$ et $\mathcal{P}^{(k)}$.*

Étape 2 : Calcul de la direction de descente en $\mathcal{V}^{(k)}$. *Si elle est nulle, passer à l'étape 3. Sinon, effectuer une recherche linéaire dans la direction $s_{\mathcal{V}}^{(k)}$ en se basant sur la fonction exacte* (9.3) *et passer à l'étape 5.*

Étape 3 : Calcul de la direction de descente en $\mathcal{U}^{(k)}$. *Si elle est nulle, passer à l'étape 4. Sinon, effectuer une recherche linéaire dans la direction $s_{\mathcal{U}}^{(k)}$ en se basant sur la fonction exacte* (9.3) *et passer à l'étape 5.*

Étape 4 : Calcul de la direction de descente en $\mathcal{W}^{(k)}$. *Utiliser $s_{\mathcal{W}}^{(k)}$ pour désactiver une contrainte de borne, modifier les espaces $\mathcal{U}^{(k)}$, $\mathcal{V}^{(k)}$ et $\mathcal{W}^{(k)}$ en conséquence et retourner à l'étape 2. S'il n'est pas possible de désactiver une contrainte de borne, stopper l'algorithme.*

Étape 5 : Évaluation des contraintes de bornes à activer. *C'est la fin de l'itération, poser $k := k+1$, retour à l'étape 1 pour l'itération suivante.*

La première étape d'une itération de base est d'identifier les *arêtes* actives. Le terme *arête* désigne les espaces de dimension inférieure à n pour lesquels une ou plusieurs fonctions $\phi_i(x)$ (pour $i = 1,\ldots,n$) sont nulles. L'évaluation de ces fonctions permet de construire les ensembles $\mathcal{N}^{(k)}$, $\mathcal{Z}^{(k)}$ et $\mathcal{P}^{(k)}$ qui reprennent les indices des fonctions $\phi_i(x)$ respectivement positives, nulles et négatives. En pratique, cette répartition est évidemment effectuée à une certaine tolérance près.

L'algorithme repose sur une exploration successive des sous-espaces $\mathcal{V}^{(k)}$, $\mathcal{U}^{(k)}$ et $\mathcal{W}^{(k)}$ en calculant les trois directions de descente $s_{\mathcal{V}}^{(k)}$, $s_{\mathcal{U}}^{(k)}$ et $s_{\mathcal{W}}^{(k)}$. Mais pourquoi dans cet ordre ? En s'inspirant des stratégies de contraintes actives, il nous est apparu plus sûr de n'autoriser la désactivation d'une arête que dans le cas où l'algorithme avait atteint un point stationnaire dans l'espace des arêtes actives. Le calcul de $s_{\mathcal{W}}^{(k)}$ qui sert à la mise à jour de l'ensemble des arêtes actives est donc effectué en dernier. Restent les sous-espaces $\mathcal{U}^{(k)}$ et $\mathcal{V}^{(k)}$. Le sous-espace $\mathcal{U}^{(k)}$ est tangent aux arêtes actives alors que le sous-espace $\mathcal{V}^{(k)}$ leur est orthogonal. Pour un itéré $x^{(k)}$ donné, dans le cas où deux directions de descente $s_{\mathcal{U}}^{(k)}$ et $s_{\mathcal{V}}^{(k)}$ sont non nulles, faut-il privilégier un déplacement tangent ou orthogonal aux arêtes actives ? La piste qui a été choisie a été la seconde. En effet, il nous est apparu moins efficace de se déplacer tangentiellement à une arête qu'il conviendra sans doute de quitter par la suite puisqu'une direction de descente orthogonale à celle-ci existe. Bien entendu, le fait que $s_{\mathcal{V}}^{(k)} = 0$ n'est en rien une garantie que l'ensemble des arêtes actives à l'optimum a été identifié mais quelques tests numériques ont confirmé notre intuition.

Exemple 9.6 *Reprenons le problème* (9.38) *à l'itéré* $x^{(0)} = (0,0,1)$*, la contrainte de borne inférieure sur la première composante est considérée comme active. Comme nous l'avons vu aux exemples 9.3, 9.4 et 9.5, les trois directions de descentes sont*

$$s_{\mathcal{U}}^{(0)} = \begin{pmatrix} 0 \\ 1/4 \\ 0 \end{pmatrix}, \qquad s_{\mathcal{V}}^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \textit{et} \qquad s_{\mathcal{W}}^{(0)} = \begin{pmatrix} 1/2 \\ 0 \\ 0 \end{pmatrix}. \tag{9.91}$$

La direction de descente en $\mathcal{V}^{(0)}$ *étant nulle, une recherche linéaire est effectuée dans la direction* $s_{\mathcal{U}}^{(0)}$*. Si la direction de descente en* $\mathcal{U}^{(0)}$ *avait également été nulle, l'examen du signe de la première composante de la direction* $s_{\mathcal{W}}^{(0)}$ *nous aurait amené à désactiver la contrainte de borne* $[x_L]_1 = 0$.

La recherche linéaire effectuée dans la direction $s_{\mathcal{U}}^{(0)}$ *aboutit à l'itéré*

$$x^{(1)} = \begin{pmatrix} 0 \\ 1/4 \\ 1 \end{pmatrix}. \tag{9.92}$$

L'ensemble $\mathcal{Z}^{(1)}$ *est vide puisque les deux fonctions* ϕ_1 *et* ϕ_2 *ont une valeur positive en* $x^{(1)}$*. La fonction approchée est dès lors*

$$\begin{aligned} \varphi^{(1)}(x) &= \phi_0(x) + \phi_1(x) + \phi_2(x) \\ &= -4 + \frac{x_1}{2} - x_2 + x_3 + 2x_2^2 + 2x_3^2. \end{aligned} \tag{9.93}$$

Puisque l'espace $\mathcal{V}^{(1)}$ *est de dimension nulle, la direction correspondante est nulle et le calcul de la direction de descente en* $\mathcal{U}^{(1)}$ *donne*

$$s_{\mathcal{U}}^{(1)} = \begin{pmatrix} 0 \\ 0 \\ -5/4 \end{pmatrix}. \tag{9.94}$$

La recherche linéaire dans cette direction aboutit à l'itéré

$$x^{(2)} = \begin{pmatrix} 0 \\ 1/4 \\ \sqrt{15}/4 \end{pmatrix} \tag{9.95}$$

qui est situé sur une seule des deux arêtes puisque $\phi_1(x^{(2)}) = 0$ *et* $\phi_2(x^{(2)}) < 0$*. Les itérations suivantes se feront tangentiellement à l'arête immédiatement suivies par un retour vers celle-ci (voir figure 9.3).*

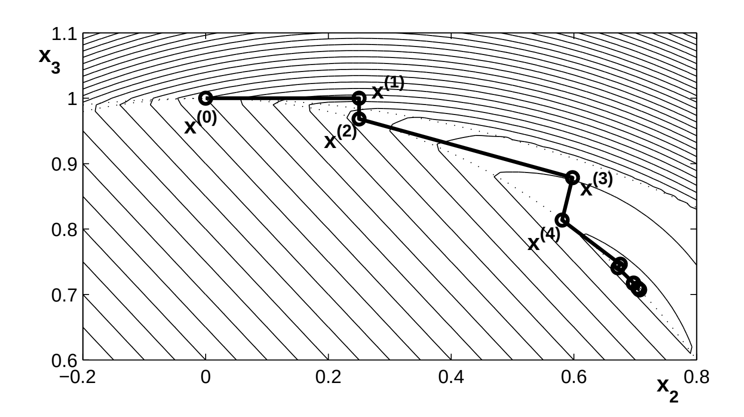


FIG. 9.3 – Illustration du comportement de l'algorithme UVQCQP de base. Représentation, dans le plan $x_1 = 0$, de la fonction (9.38). Les lignes pointillées indiquent les arêtes. La recherche linéaire effectuée dans la direction $s_{\mathcal{U}}^{(0)}$ aboutit sur l'itéré $x^{(1)}$. Puisque l'espace $\mathcal{V}^{(1)}$ est de dimension nulle, la direction de descente en $\mathcal{U}^{(1)}$ aboutit sur l'itéré $x^{(2)}$ situé sur une arête. L'itération suivante est tangentielle à celle-ci et est immédiatement suivie par une itération jouant le rôle d'une correction pour ramener l'algorithme vers la dite arête.

Les itérations qui suivent convergeront vers le point

$$x = \begin{pmatrix} 0 \\ \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} \tag{9.96}$$

pour lequel $s_{\mathcal{V}} = 0$, $s_{\mathcal{U}} = 0$ et qui est donc l'optimum[4] *dans le plan $x_1 = 0$. La direction de descente en $\mathcal{W}$, quant à elle, sera*

$$s_{\mathcal{W}} = \begin{pmatrix} 1/2 \\ 0 \\ 0 \end{pmatrix}. \tag{9.97}$$

Sa première composante est positive alors que la variable correspondante est immobilisée sur sa contrainte de borne inférieure. Dès lors, il convient de désactiver cette contrainte de borne et de recommencer une itération.

9.3.1 Calcul pratique des directions de descente.

Le calcul des différentes directions de descente est conditionné par la décomposition QR (9.29). Or cette décomposition un peu particulière dépend de la dé-

[4]En pratique, la suite d'itérés devra évidemment être interrompue lorsque les normes euclidiennes seront majorées par une certaine tolérance définie *a priori*.

composition QR de la matrice

$$\bar{G}^{(k)} = \bar{Q}^{(k)}\bar{R}^{(k)} \tag{9.98}$$

où $\bar{R}^{(k)}$ est conforme à la condition (9.28). Cette décomposition permet également de déterminer le rang de $\bar{R}^{(k)}$ qui est la dimension de l'espace $\mathcal{V}^{(k)}$. Pour rappel, la dimension de l'espace $\mathcal{U}^{(k)}$ est facilement obtenue par

$$\dim \mathcal{U}^{(k)} = n - \dim \mathcal{W}^{(k)} - \dim \mathcal{V}^{(k)} = \bar{n}^{(k)} - \dim \mathcal{V}^{(k)}. \tag{9.99}$$

Dans le calcul des différentes directions de descente, toutes les multiplications par $W^{(k)}$, $V^{(k)}$ et $W^{(k)}$ des sections précédentes sont exprimables en terme de multiplications par $\bar{Q}^{(k)}$. Ceci nous permet d'effectuer toutes les multiplications nécessaires en utilisant la décomposition de Householder de la matrice orthogonale $\bar{Q}^{(k)}$ sans avoir à construire explicitement les matrices $Q^{(k)}$ ou $\bar{Q}^{(k)}$ (voir [62]).

Par exemple, les définitions (9.49), (9.65) et (9.79) de $a_{\mathcal{U}}^{(k)}$, $a_{\mathcal{V}}^{(k)}$ et $a_{\mathcal{W}}^{(k)}$ peuvent s'écrire

$$\begin{pmatrix} a_{\mathcal{W}}^{(k)} \\ a_{\mathcal{V}}^{(k)} \\ a_{\mathcal{U}}^{(k)} \end{pmatrix} = Q^{(k)T}\left(g_0^{(k)} + \sum_{i\in\mathcal{P}^{(k)}} g_i^{(k)}\right) \tag{9.100}$$

qui peut s'exprimer par blocs

$$\begin{pmatrix} a_{\mathcal{V}}^{(k)} \\ a_{\mathcal{U}}^{(k)} \end{pmatrix} = \bar{Q}^{(k)T}\left(\bar{g}_0^{(k)} + \sum_{i\in\mathcal{P}^{(k)}} \bar{g}_i^{(k)}\right) = \bar{Q}^{(k)T}\bar{g}^{(k)} \tag{9.101}$$

et

$$a_{\mathcal{W}}^{(k)} = \tilde{g}_0^{(k)} + \sum_{i\in\mathcal{P}^{(k)}} \tilde{g}_i^{(k)} = \tilde{g}^{(k)}. \tag{9.102}$$

9.3.2 Calcul de la direction de descente en $\mathcal{V}^{(k)}$.

Si $\dim \mathcal{V}^{(k)} \neq 0$, nous calculons la direction de descente en $\mathcal{V}^{(k)}$. En tenant compte des définitions (9.68) et (9.82) des matrices $G_{\mathcal{V}}^{(k)}$ et $G_{\mathcal{W}}^{(k)}$, celles-ci apparaissent lorsqu'on effectue le produit par blocs

$$Q^{(k)T}\left(W^{(k)}\; G^{(k)}\right) = \begin{pmatrix} W^{(k)T} \\ V^{(k)T} \\ U^{(k)T} \end{pmatrix}\left(W^{(k)}\; G^{(k)}\right) = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & G_{\mathcal{W}}^{(k)} \\ 0 & G_{\mathcal{V}}^{(k)} \\ 0 & 0 \end{pmatrix}. \tag{9.103}$$

D'autre part, ce produit peut également s'écrire

$$Q^{(k)T}\left(W^{(k)}\; G^{(k)}\right) = Q^{(k)T}Q^{(k)}R^{(k)} = R^{(k)} = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & \tilde{G}^{(k)} \\ 0 & \bar{R}^{(k)} \end{pmatrix} \tag{9.104}$$

En comparant (9.103) et (9.104), il apparaît que $G_{\mathcal{W}}^{(k)} = \tilde{G}^{(k)}$ et que $G_{\mathcal{V}}^{(k)}$ peut être extraite directement de $\bar{R}^{(k)}$ puisque

$$\begin{pmatrix} G_{\mathcal{V}}^{(k)} \\ 0 \end{pmatrix} = \bar{R}^{(k)}. \tag{9.105}$$

Nous pouvons constater que les équations (9.70) et (9.72) ne demandent, pour le calcul de la direction de descente en $\mathcal{V}^{(k)}$, que le calcul des quatre produits $V^{(k)}G_{\mathcal{V}}^{(k)}$, $G_{\mathcal{V}}^{(k)T}a_{\mathcal{V}}^{(k)}$, $V^{(k)}a_{\mathcal{V}}^{(k)}$ et $G_{\mathcal{V}}^{(k)T}G_{\mathcal{V}}^{(k)}$. Le premier d'entre eux peut se développer de la manière suivante

$$\begin{aligned} V^{(k)}G_{\mathcal{V}}^{(k)} &= \begin{pmatrix} V^{(k)} & U^{(k)} \end{pmatrix} \begin{pmatrix} G_{\mathcal{V}}^{(k)} \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 \\ \bar{V}^{(k)} & \bar{U}^{(k)} \end{pmatrix} \bar{R}^{(k)} \\ &= \begin{pmatrix} 0 \\ \bar{Q}^{(k)} \end{pmatrix} \bar{R}^{(k)} = \begin{pmatrix} 0 \\ \bar{G}^{(k)} \end{pmatrix} \end{aligned} \tag{9.106}$$

et le deuxième, en tenant compte de (9.101), s'écrit

$$\begin{aligned} G_{\mathcal{V}}^{(k)T}a_{\mathcal{V}}^{(k)} &= \begin{pmatrix} G_{\mathcal{V}}^{(k)T} & 0 \end{pmatrix} \begin{pmatrix} a_{\mathcal{V}}^{(k)} \\ a_{\mathcal{U}}^{(k)} \end{pmatrix} \\ &= \bar{R}^{(k)T}\bar{Q}^{(k)T}\bar{g}^{(k)} = \bar{G}^{(k)T}\bar{g}^{(k)}, \end{aligned} \tag{9.107}$$

Le produit $V^{(k)}a_{\mathcal{V}}^{(k)}$, quant à lui, peut être développé comme suit

$$V^{(k)}a_{\mathcal{V}}^{(k)} = \begin{pmatrix} 0 \\ \bar{V}^{(k)} \end{pmatrix} a_{\mathcal{V}}^{(k)} = \begin{pmatrix} 0 \\ \bar{V}^{(k)}a_{\mathcal{V}}^{(k)} \end{pmatrix}. \tag{9.108}$$

Le bloc inférieur peut être calculé en effectuant une multiplication par $\bar{Q}^{(k)}$

$$\bar{V}^{(k)}a_{\mathcal{V}}^{(k)} = \begin{pmatrix} \bar{V}^{(k)} & \bar{U}^{(k)} \end{pmatrix} \begin{pmatrix} a_{\mathcal{V}}^{(k)} \\ 0 \end{pmatrix} = \bar{Q}^{(k)} \begin{pmatrix} a_{\mathcal{V}}^{(k)} \\ 0 \end{pmatrix}. \tag{9.109}$$

Le dernier produit $G_{\mathcal{V}}^{(k)T}G_{\mathcal{V}}^{(k)}$ est aisément obtenu en égalant la matrice (9.103) multipliée par sa transposée et la matrice (9.104) également multipliée par sa transposée

$$\begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & \tilde{G}^{(k)} \\ \tilde{G}^{(k)T} & \tilde{G}^{(k)T}\tilde{G}^{(k)} + \bar{R}^{(k)T}\bar{R}^{(k)} \end{pmatrix} = \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} & G_{\mathcal{W}}^{(k)} \\ G_{\mathcal{W}}^{(k)T} & G_{\mathcal{W}}^{(k)T}G_{\mathcal{W}}^{(k)} + G_{\mathcal{V}}^{(k)T}G_{\mathcal{V}}^{(k)} \end{pmatrix},$$

ce qui nous donne

$$G_{\mathcal{V}}^{(k)T} G_{\mathcal{V}}^{(k)} = \bar{R}^{(k)T} \bar{R}^{(k)}. \tag{9.110}$$

Le problème (9.70) peut donc s'écrire

$$l^{(k)} \in \arg\min \left(l^T \bar{G}^{(k)T} \bar{g}^{(k)} + \frac{1}{2} l^T \bar{G}^{(k)T} \bar{G}^{(k)} l \right), \tag{9.111}$$

$$\text{s. c.} \quad 0 \leq l \leq 1. \tag{9.112}$$

Constatons que ces différents calculs peuvent être effectués sans avoir à construire explicitement la matrice $\bar{Q}^{(k)}$. Si la direction de descente $s_{\mathcal{V}}^{(k)}$ est non-nulle, une recherche linéaire est effectuée pour obtenir l'itéré suivant $x^{(k+1)}$, qui servira de base à une nouvelle itération.

9.3.3 Calcul de la direction de descente en $\mathcal{U}^{(k)}$.

Si la direction $s_{\mathcal{V}}^{(k)}$ est nulle (ou si $\dim \mathcal{V}^{(k)} = 0$) et si $\dim \mathcal{U}^{(k)} \neq 0$, nous effectuons le calcul de la direction de descente en $\mathcal{U}^{(k)}$. Ceci nécessite la résolution du problème (9.51) soumis aux contraintes (9.54) et (9.55). La matrice $\bar{U}^{(k)}$ apparaissant dans ces contraintes peut être obtenue en effectuant le produit

$$\bar{Q}^{(k)} \begin{pmatrix} 0 \\ I_{\dim \mathcal{U}^{(k)}} \end{pmatrix} = \bar{U}^{(k)}. \tag{9.113}$$

La matrice $A_{\mathcal{U}}^{(k)}$, définie par l'expression (9.50), peut s'exprimer en décomposant $A^{(k)}$ par blocs

$$\begin{aligned} A_{\mathcal{U}}^{(k)} &= U^{(k)T} A^{(k)} U^{(k)} \\ &= \begin{pmatrix} 0 & \bar{U}^{(k)T} \end{pmatrix} \begin{pmatrix} \tilde{A}^{(k)} & \hat{A}^{(k)} \\ \hat{A}^{(k)T} & \bar{A}^{(k)} \end{pmatrix} \begin{pmatrix} 0 \\ \bar{U}^{(k)} \end{pmatrix} \\ &= \bar{U}^{(k)T} \bar{A}^{(k)} \bar{U}^{(k)} \end{aligned} \tag{9.114}$$

et ce dernier produit peut être extrait de l'expression

$$\bar{Q}^{(k)T} \bar{A}^{(k)} \bar{Q}^{(k)} = \begin{pmatrix} \bar{V}^{(k)T} \\ \bar{U}^{(k)T} \end{pmatrix} \bar{A}^{(k)} \begin{pmatrix} \bar{V}^{(k)} & \bar{U}^{(k)} \end{pmatrix} \tag{9.115}$$

$$= \begin{pmatrix} \bar{V}^{(k)T} \bar{A}^{(k)} \bar{V}^{(k)} & \bar{V}^{(k)T} \bar{A}^{(k)} \bar{U}^{(k)T} \\ \bar{U}^{(k)T} \bar{A}^{(k)} \bar{V}^{(k)} & \bar{U}^{(k)T} \bar{A}^{(k)} \bar{U}^{(k)} \end{pmatrix}. \tag{9.116}$$

À nouveau, ces différents calculs peuvent être effectués sans avoir à construire explicitement la matrice $\bar{Q}^{(k)}$. Si la direction de descente $s_{\mathcal{U}}^{(k)}$ est non-nulle, une recherche linéaire est effectuée pour obtenir l'itéré suivant $x^{(k+1)}$, qui servira de base à une nouvelle itération.

9.3.4 Calcul de la direction de descente en $\mathcal{W}^{(k)}$.

Si la directions $s_{\mathcal{V}}^{(k)}$ (ou si $\dim \mathcal{V}^{(k)} = 0$) est nulle, si $s_{\mathcal{U}}^{(k)}$ est également nulle (ou si $\dim \mathcal{U}^{(k)} = 0$) et si $\dim \mathcal{W}^{(k)} \neq 0$, nous calculons la direction de descente en $\mathcal{W}^{(k)}$. Dans le cas contraire, l'algorithme s'arête, aucune direction de descente n'ayant pu être trouvée.

Pour calculer cette direction $s_{\mathcal{W}}^{(k)}$, il suffit d'examiner attentivement les équations (9.83) et (9.85). Nous pouvons constater que le calcul de la direction de descente en $\mathcal{W}^{(k)}$, de façon similaire au calcul de la direction $s_{\mathcal{V}}^{(k)}$, n'exige le calcul que des quatre produits

$$\begin{aligned} W^{(k)} G_{\mathcal{W}}^{(k)} &= \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} \\ 0 \end{pmatrix} \tilde{G}^{(k)} = \begin{pmatrix} \tilde{G}^{(k)} \\ 0 \end{pmatrix}, & (9.117) \\ W^{(k)} a_{\mathcal{W}}^{(k)} &= \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} \\ 0 \end{pmatrix} \tilde{g}^{(k)} = \begin{pmatrix} \tilde{g}^{(k)} \\ 0 \end{pmatrix}, & (9.118) \\ G_{\mathcal{W}}^{(k)T} G_{\mathcal{W}}^{(k)} &= \tilde{G}^{(k)T} \tilde{G}^{(k)} & (9.119) \end{aligned}$$

et

$$\begin{aligned} G_{\mathcal{W}}^{(k)T} a_{\mathcal{W}}^{(k)} &= G^{(k)T} W^{(k)} \tilde{g}^{(k)} \\ &= \begin{pmatrix} \tilde{G}^{(k)T} & \bar{G}^{(k)T} \end{pmatrix} \begin{pmatrix} I_{\dim \mathcal{W}^{(k)}} \\ 0 \end{pmatrix} \tilde{g}^{(k)} \\ &= \tilde{G}^{(k)T} \tilde{g}^{(k)}. & (9.120) \end{aligned}$$

Ces différents calculs peuvent être effectués sans avoir à construire explicitement la matrice $\bar{Q}^{(k)}$. Si la direction de descente $s_{\mathcal{W}}^{(k)}$ est non-nulle, l'algorithme évalue si cette direction de recherche nécessite une mise à jour du caractère actif ou non des contraintes de bornes.

Si la contrainte de borne inférieure (resp. supérieure) i est *active* — ce qui implique que $[x^{(k)}]_i = [x_L]_i$ (resp. $[x^{(k)}]_i = [x_U]_i$) — et si la composante $[s_{\mathcal{W}^{(k)}}]_i$ est strictement positive (resp. strictement négative) alors cette contrainte est *candidate à la désactivation*. Parmi les contraintes candidates à la désactivation, celle dont la valeur absolue $|s_i^{(k)}|$ est la plus grande est désactivée. La désactivation simultanée de plusieurs contraintes de bornes n'est ici pas autorisée, cette technique très simple permet de se prémunir d'éventuels effets de zig-zag qui ralentissent la convergence (voir par exemple Panier [81]). L'espace $\mathcal{W}^{(k)}$ est donc modifié et le calcul des différentes directions de descente est à nouveau effectué. Dans le cas où aucune contrainte n'est candidate à la désactivation, l'algorithme stoppe et $x^{(k)}$ est accepté comme solution du problème.

L'activation de nouvelles contraintes de bornes se fait en fin d'itération, après la recherche linéaire. Il suffit d'identifier les composantes de $x^{(k)}$ qui sont égales à leur borne supérieure ou inférieure. Pour éviter de sortir du domaine admissible, l'algorithme de recherche linéaire doit donc calculer au préalable une borne maximale $\xi_U^{(k)}$ qui est évaluée en prenant le ξ correspondant au point où la direction de recherche croise la première des contraintes de bornes qu'elle rencontre (la recherche linéaire sera évoquée plus en détail dans la section 9.5).

9.4 Le mode rapide.

L'itération de base décrite dans la section précédente ne permet pas d'éviter l'*effet Maratos* décrit à la section 8.3 pour les algorithmes SQP. Pour s'en convaincre, il suffit de comparer le comportement de l'algorithme UVQCQP des figures 9.3 et 8.4, illustrant l'effet Maratos pour un algorithme de type SQP. L'effet Maratos se présente lorsque la courbure des contraintes n'est pas correctement prise en compte. Pour cette raison, en s'inspirant des corrections du second ordre utilisées pour les algorithmes SQP, un *mode rapide* a été développé.

9.4.1 Première itération en mode rapide.

Le *mode rapide* peut être activé lorsque le nombre d'arêtes actives $|\mathcal{Z}^{(k)}|$ est positif tout en étant inférieur à la dimension de travail $\overline{n}^{(k)}$. Cependant, dans un souci de simplicité, le *mode rapide* n'est pas activé lors de cas de dégénérescence des arêtes, c'est-à-dire si le nombre d'arêtes actives $|\mathcal{Z}^{(k)}|$ est inférieur à la dimension de l'espace $\mathcal{V}^{(k)}$. Enfin, l'effet Maratos ne se ressent que lorsque la direction de recherche est tangente aux contraintes (aux arêtes dans notre cas), c'est-à-dire lorsque la recherche d'une direction de descente se fait dans le sous-espace $\mathcal{U}^{(k)}$. L'ensemble de cette section prend donc comme hypothèse que la direction $s_{\mathcal{V}}^{(k)} = 0$ et que

$$0 < |\mathcal{Z}^{(k)}| = \dim \mathcal{V}^{(k)} < \overline{n}^{(k)}. \qquad (9.121)$$

Si le mode rapide est activé, nous utilisons une définition légèrement différente de la direction de descente en $\mathcal{U}^{(k)}$: le principe de calcul reste le même mais la matrice $A_{\mathcal{U}}^{(k)}$ est redéfinie selon

$$A_{\mathcal{U}}^{(k)} = U^{(k)T}\left(A_0 + \sum_{i\in\mathcal{P}^{(k)}} A_i + \sum_{i\in\mathcal{Z}^{(k)}} [z^{(k)}]_i A_i\right)U^{(k)} = U^{(k)T}A^{(k)}U^{(k)} \qquad (9.122)$$

en lieu et place de (9.50). Le vecteur $z^{(k)}$ est le vecteur des *pseudo-multiplicateurs de Lagrange*, *i.e.* les multiplicateurs de Lagrange correspondant aux arêtes actives

si celles-ci étaient des contraintes. Une estimation de ce vecteur peut être obtenu par un calcul au sens des moindre carrés

$$\hat{z}^{(k)} \in \arg\min_{z} \left\| \bar{g}^{(k)} + \bar{G}^{(k)} z \right\|^2 . \tag{9.123}$$

Sachant que $\bar{G}^{(k)} = \bar{Q}^{(k)} \bar{R}^{(k)}$ où $\bar{R}^{(k)}$ est une matrice triangulaire supérieure de rang maximum et de dimension $\bar{n}^{(k)} \times |\mathcal{Z}^{(k)}|$, le vecteur $z^{(k)}$ est la solution du système linéaire

$$\bar{R}^{(k)} \hat{z}^{(k)} = -\bar{Q}^{(k)T} \bar{g}^{(k)}. \tag{9.124}$$

Ce système peut, en tenant compte de (9.101), être exprimé par blocs

$$\begin{pmatrix} \bar{R}^{(k)}_{\mathcal{V}} \\ 0 \end{pmatrix} \hat{z}^{(k)} = - \begin{pmatrix} a^{(k)}_{\mathcal{V}} \\ a^{(k)}_{\mathcal{U}} \end{pmatrix} \tag{9.125}$$

où $\bar{R}^{(k)}_{\mathcal{V}}$ est une matrice carrée triangulaire supérieure de dimension $|\mathcal{Z}^{(k)}|$. Pour rappel, cette dernière est égale à $\dim \mathcal{V}^{(k)}$ quand le mode rapide est actif. Le calcul des pseudo-multiplicateurs de Lagrange se limite donc à l'inversion du système linéaire

$$\bar{R}^{(k)}_{\mathcal{V}} \hat{z}^{(k)} = -a^{(k)}_{\mathcal{V}}. \tag{9.126}$$

Il est intéressant de constater que le problème (9.123) peut s'écrire

$$\hat{z}^{(k)} \in \arg\min \left(z^T \bar{G}^{(k)T} \bar{g}^{(k)} + \frac{1}{2} z^T \bar{G}^{(k)T} \bar{G}^{(k)} z \right). \tag{9.127}$$

Ce dernier est équivalent au problème (9.111) dont on ne prendrait pas en compte les contraintes qui imposent à chacune des composantes de $l^{(k)}$ d'appartenir à l'intervalle $[0,1]$. Dans la suite de l'exposé, nous ferons cette approximation

$$z^{(k)} = l^{(k)} \simeq \hat{z}^{(k)} \tag{9.128}$$

où $l^{(k)}$ sont les valeurs obtenues lors de la détermination de la direction de recherche $s^{(k)}_{\mathcal{V}}$. Le fait que tous les pseudo-multiplicateurs soient contraints à être positifs a de plus l'indéniable avantage de maintenir la définie-positivité de la matrice $A^{(k)}_{\mathcal{U}}$.

La similarité entre la nouvelle définition (9.122) et la dérivée seconde du Lagrangien (8.10) utilisé dans un problème SQP n'est évidemment pas due au hasard. De façon similaire, le calcul des pseudo-multiplicateurs (9.123) est, aux notations près, identique au calcul des multiplicateurs de Lagrange d'un problème SQP classique (8.13). L'objectif est clairement de se rapprocher autant que faire

se peut des hypothèses du théorème 8.1 qui assurent une vitesse de convergence Q-quadratique.

Jusqu'à présent, rien n'empêche le pas $s_{\mathcal{U}}^{(k)}$ de souffrir de l'effet Maratos. Intuitivement, nous souhaitons donc utiliser une correction du second ordre $d^{(k)}$ semblable à celle utilisée dans les méthodes SQP. S'inspirant de la formule (8.43), celle-ci devrait être la solution du système linéaire

$$G^{(k)T} d^{(k)} = -c^{(k)} \tag{9.129}$$

où

$$c^{(k)} = \begin{pmatrix} \phi_{i_1}(x^{(k)} + s_{\mathcal{U}}^{(k)}) \\ \phi_{i_2}(x^{(k)} + s_{\mathcal{U}}^{(k)}) \\ \vdots \\ \phi_{i_{|\mathcal{Z}^{(k)}|}}(x^{(k)} + s_{\mathcal{U}}^{(k)}) \end{pmatrix} \tag{9.130}$$

avec

$$\mathcal{Z}^{(k)} = \left\{ i_1, i_2, \ldots, i_{|\mathcal{Z}^{(k)}|} \right\}. \tag{9.131}$$

Notons que chaque composante de $c^{(k)}$ peut s'écrire

$$\left[c^{(k)} \right]_j = \phi_{i_j}(x^{(k)} + s_{\mathcal{U}}^{(k)}) = \phi_{i_j}(x^{(k)}) + g_{i_j}^{(k)} s_{\mathcal{U}}^{(k)} + \frac{1}{2} s_{\mathcal{U}}^{(k)T} A_{i_j} s_{\mathcal{U}}^{(k)} \tag{9.132}$$

La définition (9.15) de l'ensemble $\mathcal{Z}^{(k)}$ permet d'affirmer que le premier terme de (9.132) est nul. Sachant que $s_{\mathcal{U}}^{(k)} = U^{(k)} u^{(k)}$, la propriété (9.36) nous permet également de conclure à l'annulation du deuxième terme de sorte que

$$\left[c^{(k)} \right]_j = \frac{1}{2} s_{\mathcal{U}}^{(k)T} A_{i_j} s_{\mathcal{U}}^{(k)}. \tag{9.133}$$

Parmi les différentes corrections du second ordre possibles, nous avons décidé de privilégier celles qui pouvaient s'exprimer $d^{(k)} = V^{(k)} d_v^{(k)}$, *i.e.* celles comprises dans le sous-espace $\mathcal{V}^{(k)}$ pour deux raisons fort simples. La première est liée aux contraintes de bornes. Restreindre la correction du second ordre $d^{(k)}$ dans le sous-espace $\mathcal{V}^{(k)}$ entraîne que le déplacement total — qui est une combinaison linéaire de $s_{\mathcal{U}}^{(k)}$ et $d^{(k)}$ — reste dans le sous-espace complémentaire à $\mathcal{W}^{(k)}$, ce qui signifie que les contraintes de bornes qui sont actives le restent et que la stratégie de contraintes actives n'est pas affectée. La deuxième raison est plus pragmatique : le système linéaire (9.129) à résoudre peut être simplifié en raison des décompositions déjà effectuées sur la matrice $G^{(k)}$ sur base des espaces $\mathcal{U}^{(k)}$ et $\mathcal{V}^{(k)}$ et

la résolution du système (9.129) en est donc simplifiée. En effet, le membre de gauche peut s'écrire par blocs

$$\begin{aligned} G^{(k)T} d^{(k)} &= G^{(k)T} V^{(k)} d_v^{(k)} && (9.134) \\ &= \left(\tilde{G}^{(k)T} \quad \bar{G}^{(k)T} \right) \left(V^{(k)} \quad U^{(k)} \right) \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} \\ &= \left(\tilde{G}^{(k)T} \quad \bar{G}^{(k)T} \right) \begin{pmatrix} 0 \\ \bar{Q}^{(k)} \end{pmatrix} \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} \\ &= \bar{G}^{(k)T} \bar{Q}^{(k)} \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} = \bar{R}^{(k)T} \bar{Q}^{(k)T} \bar{Q}^{(k)} \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} && (9.135) \\ &= \left(\bar{R}_{\mathcal{V}}^{(k)T} \quad 0 \right) \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} = \bar{R}_{\mathcal{V}}^{(k)T} d_v^{(k)} && (9.136) \end{aligned}$$

en utilisant (9.31), (9.32), (9.98) et (9.125). La correction $d^{(k)}$ est obtenue en inversant le système linéaire, inversible, carré et triangulaire inférieur

$$\bar{R}_{\mathcal{V}}^{(k)T} d_v^{(k)} = -c^{(k)} \tag{9.137}$$

puis en effectuant le produit

$$\bar{V}^{(k)} d_v^{(k)} = \left(\bar{V}^{(k)} \quad \bar{U}^{(k)} \right) \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} = \bar{Q}^{(k)} \begin{pmatrix} d_v^{(k)} \\ 0 \end{pmatrix} \tag{9.138}$$

pour obtenir

$$d^{(k)} = V^{(k)} d_v^{(k)} = \begin{pmatrix} 0 \\ \bar{V}^{(k)} \end{pmatrix} d_v^{(k)} = \begin{pmatrix} 0 \\ \bar{V}^{(k)} d_v^{(k)} \end{pmatrix}, \tag{9.139}$$

La correction du second ordre $d^{(k)}$ peut donc être calculée sans avoir à former explicitement la matrice $V^{(k)}$, au même titre que les autres opérations de ce chapitre.

Une fois construite la direction $d^{(k)}$, l'itéré suivant est simplement calculé en effectuant la minimisation unidimensionnelle suivante

$$\xi^{(k)} \in \arg\min_{\xi} \phi \left(x^{(k)} + \xi s_{\mathcal{U}}^{(k)} + \xi^2 d^{(k)} \right) \tag{9.140}$$

puis la mise à jour correspondante

$$x^{(k+1)} = x^{(k)} + \xi^{(k)} s_{\mathcal{U}}^{(k)} + (\xi^{(k)})^2 d^{(k)}. \tag{9.141}$$

9.4.2 Itérations suivantes en mode rapide.

Une fois que le mode rapide a été activé, les itérations suivantes sont également effectuées en mode rapide, jusqu'à sa désactivation (voir section 9.4.3). Supposons que l'itération $k-1$ était en mode rapide. Dans ce cas les ensembles $\mathcal{N}^{(k)}$, $\mathcal{Z}^{(k)}$ et $\mathcal{P}^{(k)}$ ne sont plus définis par les relations (9.14), (9.15) et (9.16) mais par

$$\begin{aligned}
\mathcal{N}^{(k)} &= \{j : \phi_j(x^{(k)}) < 0, j \notin \mathcal{Z}^{(k-1)}\}, && (9.142)\\
\mathcal{Z}^{(k)} &= \{j : \phi_j(x^{(k)}) = 0, j \notin \mathcal{Z}^{(k-1)}\} \cup \mathcal{Z}^{(k-1)}, && (9.143)\\
\mathcal{P}^{(k)} &= \{j : \phi_j(x^{(k)}) > 0, j \notin \mathcal{Z}^{(k-1)}\}. && (9.144)
\end{aligned}$$

Autrement dit, seules les arêtes qui n'étaient *pas* actives à l'itération précédente sont évaluées alors que celles qui étaient actives le restent. Cependant, il est fréquent qu'une arête soit activée en cours de calcul mais s'avère ne pas être active à l'optimum. Si l'algorithme ne prévoit pas de *porte de sortie* pour cette arête, elle risque d'être maintenue artificiellement à l'intérieur de l'ensemble $\mathcal{Z}^{(k)}$ et d'empêcher la convergence. C'est pourquoi à la fin de chaque itération en mode rapide — sauf si cette itération fait suite à une (ré)activation *spéciale* (voir section 9.4.4) — les *véritables* pseudo-multiplicateurs de Lagrange au sens des moindres carrés sont évalués par résolution directe du problème (9.126) au lieu d'utiliser l'approximation (9.128). Si le plus petit de ces pseudo-multiplicateurs est négatif, l'arête correspondante

$$j_a = \begin{cases} \arg\min_j \hat{z}_j & \text{si } \min_j \hat{z}_j < 0, \\ \text{indéfini} & \text{sinon,} \end{cases} \tag{9.145}$$

est retirée de l'ensemble $\mathcal{Z}^{(k-1)}$ avant d'effectuer les évaluations (9.142), (9.143) et (9.142).

Il y a seulement deux modifications à apporter par rapport à l'itération décrite dans la section précédente. Tout d'abord, la direction en $\mathcal{V}^{(k)}$ est calculée mais n'est plus utilisée tant que le mode rapide n'a pas été désactivé. L'algorithme 9.1 passe donc de l'étape 1 à l'étape 3. Ensuite, l'évaluation (9.133) des $c^{(k)}$ utilisés pour le calcul de la correction du second ordre doit être modifiée. Puisque $\phi_i(x^{(k)})$ n'est plus nécessairement nul si $i \in \mathcal{Z}^{(k-1)} \subseteq \mathcal{Z}^{(k)}$, il convient d'utiliser l'expression plus générale (9.132).

9.4.3 Désactivation du mode rapide.

Naturellement, le mode rapide doit pouvoir être *désactivé*. Plusieurs cas de figure peuvent se présenter. Le mode rapide est évidemment désactivé si les conditions (9.121) qui lui servent d'hypothèse ne sont plus satisfaites :

– si $|\mathcal{Z}^{(k)}| = 0$, *i.e.* l'ensemble $\mathcal{Z}^{(k)}$ s'avère être vide ;

- ou si $|\mathcal{Z}^{(k)}| = \bar{n}^{(k)}$, *i.e.* l'espace $\mathcal{U}^{(k)}$ est de dimension nulle ;
- ou encore si $\dim \mathcal{V}^{(k)} > |\mathcal{Z}^{(k)}|$, *i.e.* une dépendance linéaire apparaît entre les arêtes actives.

Dans tous ces cas, le mode rapide est désactivé et l'algorithme reprend son itération à la première étape. Le mode rapide est aussi désactivé si son *effet* est nul, c'est-à-dire si la correction du second ordre $d^{(k)} = 0$.

D'autre part, il peut arriver que la direction $s_{\mathcal{U}}^{(k)}$ ne soit pas une direction de descente. En effet, en raison des modifications effectuées par le mode rapide dans la manière dont elle est calculée, rien ne peut garantir que celle-ci soit bel et bien une direction de descente. Dans ce cas, le mode rapide est désactivé et l'itération relancée.

Pour éviter que l'algorithme ne s'entête à maintenir une arête active lorsqu'une direction de descente importante se présente dans l'espace $\mathcal{V}^{(k)}$, un gardien heuristique a été implémenté. Si $\|s_{\mathcal{V}}^{(k)}\| > \|s_{\mathcal{U}}^{(k)}\|$ et qu'aucune désactivation d'arête régulière n'est prévue[5], la direction de descente en $\mathcal{V}^{(k)}$ est considérée comme *dominante,* le mode rapide est stoppé et une recherche linéaire est effectuée dans la direction $s_{\mathcal{V}}^{(k)}$.

Il peut également arriver que la direction $s_{\mathcal{U}}^{(k)}$ et que la direction $s_{\mathcal{V}}^{(k)}$ soient nulles sans pour autant que le point $x^{(k)}$ soit un véritable minimum dans le sous-espace $\mathcal{W}^{(k)\perp}$. Avant de désactiver une contrainte de borne, l'algorithme désactive donc le mode rapide et reprend l'itération au départ pour vérifier qu'il ne s'agit pas là d'un *mauvais cas du mode rapide*.

Enfin, pour éviter une interaction malheureuse entre la stratégie de contraintes actives pour les contraintes de bornes et le mode rapide, ce dernier est désactivé aussitôt qu'une nouvelle contrainte de borne est activée.

9.4.4 Activation spéciale du mode rapide.

Quelques expériences numériques ont montré que des phénomènes semblables au *zigzaging* rencontré en optimisation contrainte pouvaient apparaître. Ceux-ci ralentissent considérablement les performances de l'algorithme. En effet, celui-ci oscille d'une arête à l'autre parce que la direction $s_{\mathcal{V}}^{(k)}$ est non-nulle sur chacune de ces arêtes alors que l'optimum réel se situe à leur intersection. C'est pour pallier à ce problème qu'une *activation spéciale* du mode rapide a été introduite.

L'activation spéciale a lieu si $|\mathcal{Z}^{(k-1)}| = 0$ et si une arête est activée suite à une recherche unidimensionnelle. Dans ce cas, le mode rapide est immédiatement activé et cette arête ne pourra être désactivée au cours de cette itération. De la même manière, on parle d'une *réactivation spéciale* lorsqu'une arête, inactive

[5] Aucune désactivation d'arête régulière n'est prévue si j_a défini par (9.145) est indéfini.

jusque là, doit être activée suite à la recherche unidimensionnelle. Dans ce cas, aucune arête ne peut être désactivée au cours de cette itération.

Algorithme 9.2 *Le schéma complet d'une itération s'écrit donc schématiquement comme suit. La variable entière j_a est indéfinie au début de l'algorithme. La variable logique* SPECIAL *est initialisée à 0 au début de chaque itération (avant l'étape 1).*

Étape 1 : *Identification des arêtes actives.*

Si le mode rapide n'est pas activé :

1. *utiliser les formules* (9.14), (9.15) *et* (9.14) *pour construire les ensembles $\mathcal{N}^{(k)}$, $\mathcal{Z}^{(k)}$ et $\mathcal{P}^{(k)}$;*
2. *si $|\mathcal{Z}^{(k-1)}| = 0$ et $|\mathcal{Z}^{(k)}| \neq 0$, poser* SPECIAL $:= 1$*, activer le mode rapide et recommencer l'étape 1. Sinon, passer à l'étape 2.*

Si le mode rapide est activé :

1. *si* SPECIAL $= 0$ *et j_a défini, poser $\mathcal{Z}^{(k-1)} := \mathcal{Z}^{(k-1)}/\{j_a\}$;*
2. *utiliser les mises à jour* (9.142), (9.143) *et* (9.142) *pour construire les ensembles $\mathcal{N}^{(k)}$, $\mathcal{Z}^{(k)}$ et $\mathcal{P}^{(k)}$;*
3. *si une nouvelle arête est activée, poser* SPECIAL $:= 1$ *et passer à l'étape 2.*

Étape 2 : *Calcul de la direction de descente en $\mathcal{V}^{(k)}$.*

Si le mode rapide n'est pas activé : si $s_{\mathcal{V}}^{(k)} = 0$, passer à l'étape 3. Sinon, effectuer une recherche linéaire dans cette direction et passer à l'étape 5.

Si le mode rapide est activé : si $|\mathcal{Z}^{(k)}| = 0$, $|\mathcal{Z}^{(k)}| = \bar{n}^{(k)}$ ou $|\mathcal{Z}^{(k)}| \neq \dim \mathcal{V}^{(k)}$, désactiver le mode rapide et recommencer l'étape 1. Sinon, passer à l'étape 3.

Étape 3 : *Calcul de la direction de descente en $\mathcal{U}^{(k)}$.*

Si le mode rapide n'est pas activé :

1. *si $0 < |\mathcal{Z}^{(k)}| = \dim \mathcal{V}^{(k)} < \bar{n}^{(k)}$, activer le mode rapide et recommencer l'étape 3 ;*
2. *si $s_{\mathcal{U}}^{(k)} = 0$, passer à l'étape 4. Sinon, effectuer une recherche linéaire dans cette direction et passer à l'étape 5.*

Si le mode rapide est activé :

1. *le calcul de la direction $s_{\mathcal{U}}^{(k)}$ s'effectue avec la matrice* (9.122) *au lieu de* (9.114) *;*
2. *si $s_{\mathcal{U}}^{(k)} = 0$, désactiver le mode rapide et recommencer l'étape 1 ;*

3. *si* SPECIAL $= 0$*, évaluer* j_a *avec la formule* (9.145)*, sinon poser* j_a *indéfini ;*
4. *si* $s_{\mathcal{V}}^{(k)}$ *est dominante, i.e.* $\|s_{\mathcal{V}}^{(k)}\| > \|s_{\mathcal{U}}^{(k)}\|$*, désactiver le mode rapide et recommencer l'étape 1 ;*
5. *calculer la correction du second ordre* $d^{(k)}$ *;*
6. *si* $d^{(k)} = 0$*, désactiver le mode rapide ;*
7. *effectuer une recherche unidimensionnelle du type* (9.140) *;*
8. *si le pas* $\xi^{(k)}$ *obtenu est strictement positif, calculer l'itéré suivant par* (9.141) *et passer à l'étape 5. Dans le cas contraire, désactiver le mode rapide et recommencer l'étape 1.*

Étape 4 : *Calcul de la direction de descente en* $\mathcal{W}^{(k)}$*. Utiliser* $s_{\mathcal{W}}^{(k)}$ *pour désactiver une contrainte de borne, modifier les espaces* $\mathcal{U}^{(k)}$*,* $\mathcal{V}^{(k)}$ *et* $\mathcal{W}^{(k)}$ *en conséquence et retourner à l'étape 2. S'il n'est pas possible de désactiver une contrainte de borne, stopper l'algorithme. Noter que l'étape 4 n'est jamais atteinte en mode rapide.*

Étape 5 : *Évaluer les contraintes de bornes à activer. Si une nouvelle contrainte de borne est activée, désactiver le mode rapide si celui-ci était activé. Fin de l'itération, poser* $k := k+1$*, retour à l'étape 1 pour l'itération suivante.*

9.4.5 Performances

L'introduction du mode rapide complique singulièrement l'algorithme : c'est assez facile à constater en comparant les algorithmes 9.1 et 9.2. Le jeu en vaut cependant la chandelle. L'intérêt du mode rapide est de contrer l'effet Maratos par l'introduction d'une correction du second ordre (voir section 8.3). L'efficacité du mode rapide peut être clairement illustrée par l'exemple suivant.

Exemple 9.7 *Reprenons la fonction* (9.38) *de l'exemple 9.1 au point* $x^{(0)} = (0,0,1)$*. Comme précédemment, supposons que la contrainte de borne inférieure pour la troisième variable* x_1 *est active, i.e.* $[x_L]_1 = 0$*. La figure 9.4 illustre le comportement des premières itérations des algorithmes 9.1 et 9.2 dans le plan* $x_1 = 0$*.*

Pour les deux premières itérations, le comportement des deux algorithmes est identique car le mode rapide n'est pas activé (voir exemple 9.6). En $x^{(0)}$*, bien que des arêtes soient actives, le mode rapide n'est pas enclenché puisque le nombre d'arêtes actives* $|\mathcal{Z}^{(0)}| = 2$ *n'est pas égal à la dimension de l'espace* $\mathcal{V}^{(0)}$ *(*$\dim \mathcal{V}^{(0)} = 1$*). L'algorithme a ainsi détecté une dépendance linéaire du premier ordre entre les arêtes et n'a pas tenté de « suivre la courbure » des deux*

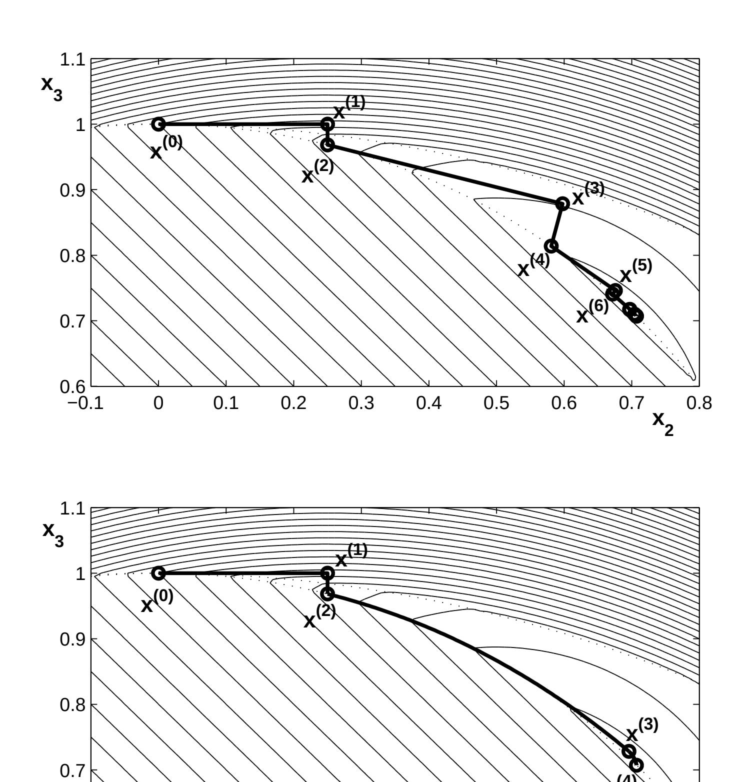


FIG. 9.4 – Comportement des premières itérations de l'algorithme UVQCQP dans le plan $x_1 = 0$. La figure du dessus présente le comportement sans mode rapide et celle du dessous le comportement avec déclenchement du mode rapide (voir exemple 9.7). Sur la première, les recherches unidimensionnelles entre les itérés sont *linéaires*. Sur la seconde, ces mêmes recherches prennent la forme d'une trajectoire *parabolique* en cas d'activation du mode rapide.

arêtes puisque cette opération est impossible. En $x^{(1)}$*, il n'y a pas d'arête active* ($|\mathcal{Z}^{(1)}| = 0$) *et le mode rapide ne peut donc être activé.*

C'est à partir de l'itéré $x^{(2)}$ *que les comportements diffèrent. En effet, en ce point, le nombre d'arêtes actives est égal à la dimension de l'espace* $\mathcal{V}^{(0)}$ *et le mode rapide peut dès lors être activé. La recherche unidimensionnelle suit alors une trajectoire parabolique et non plus linéaire. Pour les deux itérations suivantes, l'ensemble des arêtes actives* $\mathcal{Z}^{(k)}$ *reste inchangé. Après cinq itérations, les directions de recherche en* $\mathcal{U}$ *et en* $\mathcal{V}$ *sont nulles (en réalité leur norme euclidienne est inférieure à* 10^{-5}*). L'itéré* $x^{(5)}$ *est donc l'optimum dans le plan* $x_1 = 1$*. En l'absence de mode rapide, ce résultat est atteint en quinze itérations.*

9.5 Minimisation unidimensionnelle.

Lorsqu'une direction de descente $s^{(k)}$ et, éventuellement, une correction du second ordre $d^{(k)}$ ont été calculées, une minimisation unidimensionnelle est effectuée le long de la « trajectoire parabolique »

$$x^{(k+1)} = x^{(k)} + \xi^{(k)} s^{(k)} + (\xi^{(k)})^2 d^{(k)}. \tag{9.146}$$

Si la direction $d^{(k)}$ est nulle, la « trajectoire » dégénère en une ligne droite et la minimisation unidimensionnelle en une recherche linéaire classique. Dans un soucis de simplicité, les indicateurs d'itération portés en exposant seront omis dans la suite de cette section.

Chaque fonction individuelle $\phi_i(x)$ peut être développée grâce à (9.146) et nous pouvons définir les fonctions unidimensionnelles

$$\begin{aligned}
\psi_i(\xi) &= \phi_i(x+\xi s+\xi^2 d) \\
&= \alpha_i + a_i^T x + \frac{1}{2} x^T A_i x + (a_i^T s + x^T A_i s)\xi \\
&\quad + (a_i^T d + x^T A_i d + + \frac{1}{2} s^T A_i s)\xi^2 + d^T A_i s \xi^3 + \frac{1}{2} d^T A_i d \xi^4 \\
&= \delta_i + \beta_i \xi + \gamma_i \xi^2 + \kappa_i \xi^3 + \iota_i \xi^4
\end{aligned} \tag{9.147}$$

pour $i = 0, \ldots, m$. Toutes ces fonctions sont des polynômes de degré quatre et la fonction à minimiser correspondante

$$\psi(\xi) = \psi_0(\xi) + \sum_{i=1}^{m} \max(0, \psi_i(\xi)) \tag{9.148}$$

est dès lors une fonction non dérivable continue égale, par morceaux, à des polynômes de degré quatre.

Il existe de nombreux algorithmes tout à fait généraux pour effectuer la minimisation de cette fonction non-différentiable (voir par exemple [64]) mais la forme particulière de ce problème correspond à ceux résolus avec une très grande précision par Murray et Overton [76]. Toutefois, ces auteurs envisagent la minimisation de fonctions du type (9.148) avec $\psi_0(\xi) = 0$ et des fonctions générales $\psi_i(\xi)$ continûment différentiables. Étant donné le caractère polynomial des fonctions ψ_i et la présence ψ_0, une version simplifiée, plus efficace et plus précise encore a été spécialement développée dans le cadre de ce travail.

9.5.1 Intervalle de confiance

Nous avons vu dans la section 2.1 que les méthodes les plus efficaces de minimisation unidimensionnelle font appel à un *intervalle de confiance* $[\xi_g, \xi_d]$ dans lequel se trouve le minimum que nous cherchons. Dans le cas qui nous occupe, l'intervalle initial prend évidemment comme *borne de gauche* $\xi_g = 0$. La borne de droite, quant à elle, est évaluée en déterminant la première contrainte de borne croisée par la trajectoire parabolique (9.146). Nous définissons l'ensemble des valeurs de ξ qui réalisent ces intersections

$$\begin{aligned} \Xi &= \left\{\xi : [x]_i + \xi[x]_i + \xi^2[d]_i = [x_U]_i\right\} && (9.149) \\ &\cup \left\{\xi : [x]_i + \xi[x]_i + \xi^2[d]_i = [x_L]_i\right\}, && (9.150) \end{aligned}$$

dont les éléments sont facilement identifiés par résolution des équations du second ordre correspondantes. La *borne de droite* initiale ξ_d est le plus petit élément positif de cet ensemble

$$\xi_M = \min_{\xi > 0} \xi \in \Xi. \qquad (9.151)$$

Ceci fait en sorte que, quelle que soit la valeur obtenue à la fin de la minimisation unidimensionnelle, l'itéré suivant soit bien admissible.

9.5.2 Recherche des points anguleux.

La première étape de l'algorithme recense tous les zéros des fonctions ψ_i auxquels la fonction ψ_i change de signe pour $i = 1, \ldots, m$. Ces zéros sont au nombre de $4m$ au maximum. Ils peuvent être calculés au moyen des formules exactes pour les polynômes d'ordre inférieur ou égal à quatre (voir annexe B). La plupart d'entre eux sont des poins *anguleux*[6], ce qui signifie que la fonction y est continue mais pas dérivable, et que leur représentation graphique présente donc généralement un angle en ce point.

[6]Il n'y a que dans le cas où un zéro s'avère être multiple que le point n'est pas anguleux.

Considérons donc l'ensemble des points anguleux ζ_j pour $j = 1, \ldots, p$ tels que

$$\psi_i(\zeta_j) = 0 \tag{9.152}$$

pour au moins un $i = 0, \ldots, m$. Parmi ces points[7], retenons ceux qui sont strictement positifs tout en étant inférieurs à la valeur maximum ξ_M et ordonnons-les de sorte que

$$0 < \zeta_1 < \zeta_2 < \ldots < \zeta_p \leq \xi_M. \tag{9.153}$$

9.5.3 Algorithme de minimisation.

Si $p = 0$, il n'y a aucun point anguleux dans l'intervalle $[0, \xi_M]$ et la fonction $\psi(\xi)$ est égale, dans tout cet intervalle, à un polynôme du quatrième ordre. Son minimum dans $[0, \xi_M]$ est aisément calculé en utilisant les méthodes de résolution des équations du troisième ordre pour annuler sa dérivée.

Si $p \neq 0$, nous utilisons l'algorithme itératif suivant.

Algorithme 9.3 *Initialiser le* curseur i_c *à* 0.

Étape 1. *Y a-t-il encore des points anguleux dans l'intervalle de confiance ? Si $i_c > p$, stopper l'algorithme : ξ_t est le minimum recherché. Sinon, passer à l'étape 2.*

Étape 2. *Y a-t-il un minimum avant le prochain point anguleux ? Si $i_c \leq p$ et $\xi_t \leq \zeta_{i_c}$, stopper l'algorithme : ξ_t est le minimum recherché. Sinon, passer à l'étape 3.*

Étape 3. *Réduire l'intervalle de confiance. Poser $\xi_g := \zeta_{i_c}$ et $i_c := i_c + 1$ et passer à l'étape 4.*

Étape 4. *Construire un polynôme à minimiser. Si $i_c \leq p$, poser*

$$\hat{\xi} := \frac{\zeta_{i_c-1} + \zeta_{i_c}}{2}, \tag{9.154}$$

sinon, poser

$$\hat{\xi} := \frac{\zeta_{i_c-1} + \xi_d}{2} \tag{9.155}$$

pour évaluer les coefficients du polynôme du quatrième ordre égal à $\psi(\xi)$ au voisinage de ξ_m

$$\begin{pmatrix} \hat{\delta} \\ \hat{\beta} \\ \hat{\gamma} \\ \hat{\kappa} \\ \hat{\iota} \end{pmatrix} := \begin{pmatrix} \delta_0 \\ \beta_0 \\ \gamma_0 \\ \kappa_0 \\ \iota_0 \end{pmatrix} + \sum_{\substack{i=1 \\ \psi_i(\hat{\xi}) \geq 0}}^{m} \begin{pmatrix} \delta_i \\ \beta_i \\ \gamma_i \\ \kappa_i \\ \iota_i \end{pmatrix}. \tag{9.156}$$

[7] Si la multiplicité d'un zéro correspondant à ψ_i est double ou quadruple, le point est écarté de l'ensemble des points anguleux car la fonction ψ_i n'y change pas de signe.

Passer à l'étape 5.

Étape 5. *Minimiser le polynôme. Le minimum ξ_t du polynôme*

$$\hat{\delta}+\hat{\beta}\xi+\hat{\gamma}\xi^2+\hat{\kappa}\xi^3+\hat{\iota}\xi^4 \tag{9.157}$$

dans l'intervalle $[\xi_g,\xi_d]$ est aisément calculé en utilisant les méthodes de résolution exacte pour les équations du troisième degré pour la recherche des zéros de sa dérivée. Retourner à l'étape 1.

9.6 Performances de l'algorithme.

Pour étudier les performances de l'algorithme UVQCQP, nous avons utilisé une série de problèmes de type (9.3) générés semi-aléatoirement (de manière à ce qu'ils soient reproductibles). La distribution de la dimension n du problème était répartie entre 2 et 22 tandis que le nombre de contraintes m était distribuée entre 1 et 21, *i.e.*

$$\begin{aligned} n &= 2+I_n, & (9.158)\\ m &= 1+I_m & (9.159)\end{aligned}$$

où I_m et I_n sont des nombre entiers aléatoires uniformément répartis entre 0 et 20. Les tests ci-après ne portent que sur des problèmes de petite taille car l'algorithme UVQCQP n'a pas été conçu pour prendre en charge des problèmes de grande taille : pour une utilisation à plus grande échelle, il conviendrait probablement d'effectuer quelques adaptations pour réduire l'espace mémoire utilisé par les différentes matrices et, de la sorte, le nombre d'opérations.

Les paramètres du problème (9.3) ont également été générés aléatoirement pour prendre des valeurs situées entre -600 et 600, i.e.

$$\begin{aligned} -600 \leq \quad & \alpha_i \quad \leq 600, & (9.160)\\ -600 \leq \quad & [a_i]_j \quad \leq 600, & (9.161)\\ -600 \leq \quad & [A_i]_{j,k} \quad \leq 600 & (9.162)\end{aligned}$$

pour tout $i=0,\ldots,m$ et pour tout $j,k=1,\ldots,n$. La matrice hessienne A_i des fonctions $\varphi_i(x)$ est évidemment construite de manière à ce que la symétrie

$$[A_i]_{j,k}=[A_i]_{k,j} \tag{9.163}$$

soit respectée. Cette génération aléatoire ne garantit toutefois pas la semi-définie positivité de la matrice A_i. Pour pallier à ce problème, une diagonalisation de

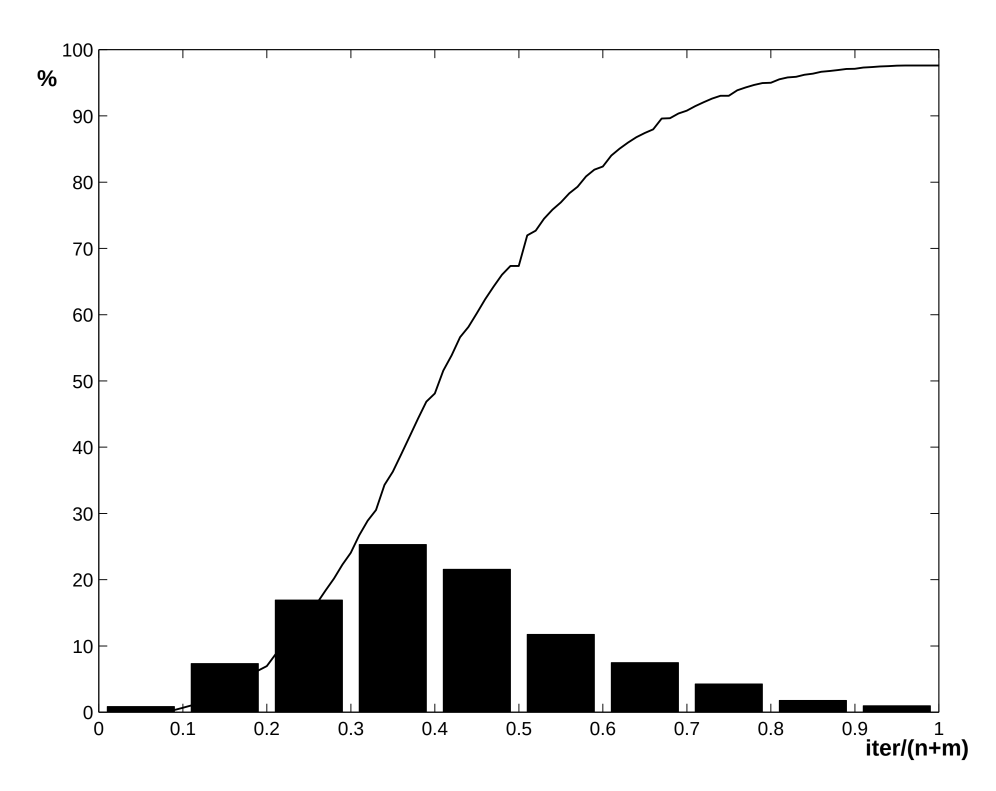


FIG. 9.5 – Performances de l'algorithme UVQCQP sur 100.000 problèmes de petite taille. L'histogramme indique la proportion des problèmes résolus en fonction de leur rapport de performance. La courbe continue représente quant à elle la proportion $p(\tau)$ de problèmes dont le rapport de performance est inférieur ou égal à τ.

la matrice est effectuée et les valeurs propres négatives sont remplacées par une valeur propre nulle (voir la formule (3.35))

$$A_i = Q^{(k)} \operatorname{diag}[\max(\lambda_i, 0)] \, Q^{(k)T}. \tag{9.164}$$

La matrice ainsi générée est alors semi-définie positive et le problème est alors bien convexe. Notons au passage que la probabilité que les matrices A_i possèdent une ou plusieurs valeur(s) propre(s) nulle(s) est non négligeable : les problèmes ainsi générés comprennent donc une proportion importante de ce cas de figure.

Enfin, les composantes du point de départ $x^{(0)}$ ont également été aléatoirement générées entre -600 et 600, tout comme les composantes des contraintes de bornes x_L et x_U. Toutefois, s'il s'avérait que le nombre généré pour la borne inférieure $[x_L]_i$ était supérieur à la composante correspondante $[x^{(0)}]_i$ de l'itéré de départ, la borne inférieure était alors modifiée en prenant $[x_L]_i = [x^{(0)}]_i$. Un raisonnement similaire s'appliquait aux bornes supérieures.

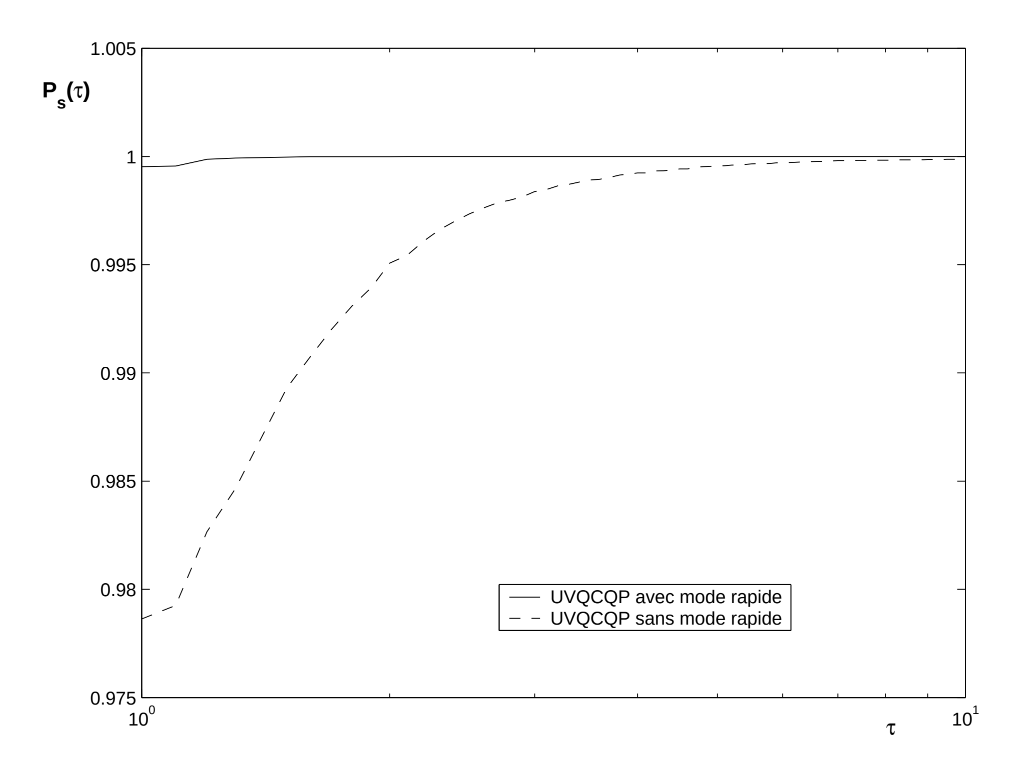


FIG. 9.6 – Profil de performance de l'algorithme UVQCQP avec et sans mode rapide sur 100.000 problèmes tests de petite taille($2 \leq n \leq 22$ et $1 \leq m \leq 21$).

La figure 9.5 présente une illustration des résultats obtenus sur 100.000 problèmes de petite taille. Comme mesure de performance, nous utilisons le rapport

$$\frac{N_i}{n+m} \tag{9.165}$$

où N_i est le nombre d'itérations nécessaire pour obtenir $\|s^{(k)}\| \leq 10^{-5}$. Ce rapport a été choisi car il met en balance la charge de calcul (le nombre d'itérations) et la complexité du problème (la dimension et le nombre de contraintes). Nous pouvons constater que 98% des problèmes sont résolus en moins de $n+m$ itérations ; ce pourcentage monte à $99,5\%$ pour un rapport de performance de 3 et les 100% sont pratiquement atteints autour de 10.

Rappelons qu'en utilisant comme mesure de performance le nombre d'itérations $N_{p,s}$ nécessaires pour résoudre le problème p avec le solveur s, nous pouvons tracer le *profil de performance* d'un solveur s. Il s'agit de la distribution cumulée du rapport de performance (7.16). Nous allons nous attarder sur quelques profils de performance.

La figure 9.6 dresse le profil obtenu sur 100.000 problèmes générés aléatoirement en présence et en l'absence du dispositif de *mode rapide*. Nous pouvons

constater que dans un peu plus 97,5% des problèmes générés, le mode rapide n'entre pas en action et les performances des deux versions de l'algorithme sont dès lors rigoureusement identiques. Il n'y a donc que 2184 cas pour lesquels le mode rapide s'active. Sur ceux-ci, 2137 sont à l'avantage de l'utilisation du mode rapide qui peut se réveler jusqu'à 40 fois plus rapide. Paradoxalement, 47 problèmes s'avèrent être légèrement plus lents en présence du mode rapide : le cas le plus problématique engendre le double d'itérations, tous les autres entraînent un surcroît d'itérations inférieur à 60%. L'introduction du mode rapide permet donc un gain de performance considérable dans plus de 2% des problèmes envisagés et une légère dépréciation dans 0,005%.

Notons également que les dispositifs de désactivation du mode rapide ou des arêtes (voir section 9.4), s'ils peuvent paraître alambiqués n'en sont pas moins indispensable au niveau de la robustesse et de la fiabilité de l'algorithme. Pour analyser leur utilité, les mêmes 100.000 problèmes générés aléatoirement ont été résolus par l'algorithme UVQCQP en l'absence de ces dispositifs. La convergence n'est pas efficacement pour 780 de ces 100.000 problèmes (le rapport de performance (9.165) était nettement supérieur à 10). Dans le cas où tous les dispositifs sont présents, plus aucun ne présente ce souci.

Nous avons également comparé l'algorithme UVQCQP avec un algorithme standard du logiciel Matlab. Le problème (9.3) ayant une forme particulière, nous avons sélectionné l'algorithme qui appréhendait le mieux cette structure. Il s'agit d'une méthode dite de « point intérieur » utilisée pour résoudre le problème contraint

$$\begin{aligned} \text{minimiser} \quad & \phi_0(x) \\ \text{s.c.} \quad & \phi_i(x) \leq 0 \qquad \text{pour } i = 1, \ldots, m. \\ \text{et} \quad & x_L \leq x \leq x_U. \end{aligned} \tag{9.166}$$

Pour être précis, la fonction utilisée était fmincon de la version *Matlab Release 2009a*. Parmi les trois algorithmes utilisables, nous avons choisi la méthode de point intérieur car celle-ci permettait d'introduire des informations du second ordre sur les contraintes comme c'est le cas pour l'algorithme UVQCQP. Malheureusement, la fonction de mérite utilisée par Matlab ne correspond pas exactement à la fonction objectif que nous utilisons tout au long de ce chapitre. La comparaison des résultats est donc à envisager avec prudence : elle n'a pour seul objectif de nous donner un *ordre de grandeur* pour comparer la charge de calcul. La figure 9.7 dresse le profil de performance ainsi obtenu sur 500 problèmes générés aléatoirement. Nous pouvons constater que ces ordres de grandeur sont nettement à l'avantage de l'algorithme UVQCQP. Fort heureusement, notre méthode *ad hoc* — *i.e.* spécialement élaborée pour résoudre des problèmes du type (9.3) — s'avère plus performante qu'un algorithme général.

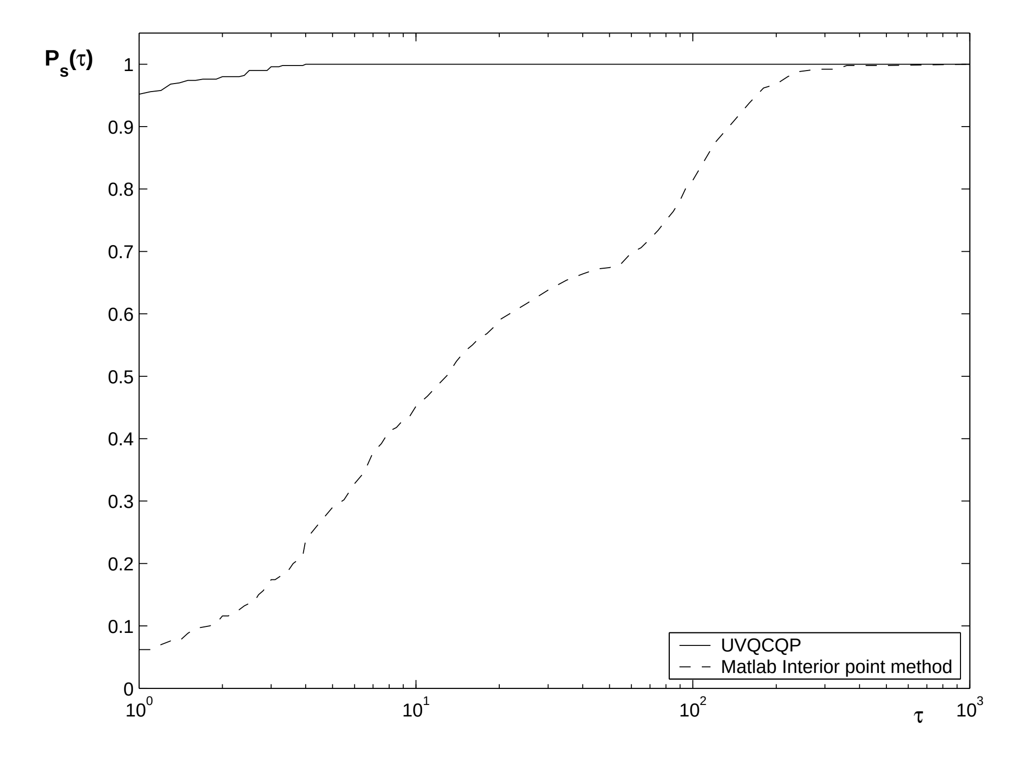


FIG. 9.7 – Profil de performance de l'algorithme UVQCQP et de la méthode de point intérieur de Matlab sur 500 problèmes tests de petite taille ($2 \leq n \leq 22$ et $1 \leq m \leq 21$).

9.7 Conclusion.

L'algorithme développé dans ce chapitre est une « brique élémentaire » pour la résolution de sous-problèmes engendrés par une approche séquentielle de problèmes quadratiques à contraintes quadratiques. Les problèmes envisagés sont ceux obtenus par l'utilisation d'une fonction de pénalité de type ℓ_1 pour une fonction objectif et des contraintes quadratiques et une région de confiance utilisant une norme ℓ_∞. Nous n'avons cependant envisagé que des sous-problèmes convexes. La fonction ainsi obtenue est donc convexe, non-différentiable, quadratique par morceaux et soumise à des contraintes de bornes.

L'algorithme UVQCQP s'inspire d'un développement théorique proposé par Lemaréchal *et al.* [65]. L'espace d'optimisation est divisé en trois sous-espaces : celui des contraintes de bornes actives $\mathcal{W}$ d'une part et son complément orthogonal $\mathcal{W}^\perp$ qui est lui-même décomposé en deux sous-espaces $\mathcal{U}$ et $\mathcal{V}$. L'espace $\mathcal{U}(x)$ est le plus grand sous-espace de $\mathcal{W}^\perp$ dans lequel la fonction est différentiable au voisinage de x. L'algorithme proposé calcule trois directions de descente dans chacun des sous-espaces et les utilise, dans un ordre précis, pour effectuer des recherches linéaires.

L'algorithme UVQCQP tient compte des développements utilisés pour accélérer les méthodes SQP. Le *mode rapide* utilise notamment la technique de la correction du second ordre et s'inspire également des stratégies de contraintes actives. Une méthode de minimisation unidimensionnelle adaptée à la structure du problème à également été développée.

Chapitre 10

Vers une méthode SQCQP — Sequential Quadratically Constrained Quadratic Programming

Ce chapitre présente les grandes lignes de l'utilisation de l'algorithme UV-QCQP développé dans le chapitre précédent pour implémenter une méthode séquentielle de programmation quadratique à contraintes quadratiques (SQCQP). Contrairement aux méthodes de type SQP abordées dans le chapitre 8, les méthodes SQCQP résolvent, à chaque itération, un sous-problème impliquant une fonction objectif quadratique et des contraintes quadratiques.

Dans ce chapitre, nous abordons succinctement les problèmes principaux rencontrés dans les méthodes de type SQCQP : la faisabilité du sous-problème, la convergence globale, le taux de convergence, la non-convexité des contraintes et la présence de contraintes d'égalité. Il ne s'agit pas d'un exposé approfondi mais seulement d'une ébauche d'utilisation des concepts introduits tout au long de ce travail. Le lecteur intéressé est invité à consulter les travaux de Kruk et Wolkowicz [58], Fukushima *et al.* [37] ou Jian [53, 54]. L'originalité de notre méthode réside dans l'utilisation des régions de confiance et de l'algorithme UVQCQP pour la résolution des sous-problèmes générés.

Nous envisageons donc le problème d'optimisation non-linéaire

$$\begin{array}{rll} \text{minimiser} & f(x), & \\ \text{s.c.} & c_j(x) = 0 & \text{pour } j \in \mathcal{E}, \\ & c_j(x) \leq 0 & \text{pour } j \in \mathcal{I}, \\ & x_L \leq x \leq x_U & \end{array} \tag{10.1}$$

où $\mathcal{E}$ et $\mathcal{I}$ sont, respectivement, les ensembles disjoints des indices des contraintes

d'égalité et d'inégalité. Les fonctions f et c_i sont supposées deux fois continûment dérivables. Les vecteurs x_L et x_U sont des contraintes de bornes sur les variables x. Pour caractériser ce problème, nous utilisons la fonction de mérite (8.15)

$$\Psi(x,\sigma) = f(x) + \sigma \sum_{j\in\mathcal{E}} |c_j(x)| + \sigma \sum_{j\in\mathcal{I}} \max[0, c_j(x)]. \qquad (10.2)$$

Les contraintes de bornes $x_L \leq x \leq x_U$ ne sont pas intégrées à Ψ. Le problème prend donc la forme d'une minimisation quasi non-contrainte d'une fonction non-différentiable. Notons que $\Psi(x,\sigma)$ est continue au sens de Lipschitz et régulière sur $\mathbb{R}^n$ (hypothèse 4.14).

10.1 Algorithme de base.

Nous allons résoudre ce problème avec une méthode par régions de confiance et résoudre les sous-problèmes créés grâce à l'algorithme UVQCQP développé au chapitre 9.

À chaque itération, nous souhaitons utiliser des approximations quadratiques convexes, aussi bien pour les contraintes que pour la fonction objectif. Pour ce faire, les contraintes du problème (10.1) sont utilisées sous la forme équivalente

$$\begin{array}{rcll} c_j(x) & \leq & 0 & \text{pour } j \in \mathcal{E}, \\ -c_j(x) & \leq & 0 & \text{pour } j \in \mathcal{E}, \\ c_j(x) & \leq & 0 & \text{pour } j \in \mathcal{I}. \end{array} \qquad (10.3)$$

L'algorithme proposé ici est volontairement extrêmement simple : notons que l'étude complète et systématique de ses propriétés théoriques et de ses performances constitue un prolongement naturel de ce travail. Notons que l'algorithme décrit ci-dessous ne comporte pas de critère d'arrêt. En pratique, nous avons choisi de stopper l'algorithme lorsque $\|s^{(k)}\|_\infty \leq 10^{-6}$.

Algorithme 10.1 *Soit un point de départ $x^{(0)}$, un rayon de confiance initial $\Delta^{(0)}$ et les constantes*

$$\begin{array}{ll} \eta_1 = 0,01\,; & \alpha_1 = 0,8\,; \\ \eta_2 = 0,95\,; & \alpha_2 = 2\,; \\ \eta_3 = 1,05\,; & \alpha_3 = 1,26 \end{array} \qquad (10.4)$$

qui satisfont aux conditions (2.28) *et* (4.53). *Calculer $\Psi(x^{(0)},\sigma)$ et initialiser $k = 0$.*

Étape 1 : Définition de l'approximation locale. *Approcher la fonction* $\Psi(x,\sigma)$ *par*

$$\begin{aligned} m^{(k)}(x,\sigma) &= \check{f}^{(k)}(x) + \sigma \sum_{j\in\mathcal{I}\cup\mathcal{E}} \max[0,\check{c}_j^{(k)}(x)] \\ &\quad + \sigma \sum_{j\in\mathcal{E}} \max[0,\hat{c}_j^{(k)}(x)] \end{aligned} \tag{10.5}$$

où $\check{f}(x^{(k)}+s)$, $\check{c}_j(x^{(k)}+s)$ *et* $\hat{c}_j(x^{(k)}+s)$ *sont des approximations quadratiques convexes de* $f(x^{(k)}+s)$, $c_j(x^{(k)}+s)$ *et* $-c_j(x^{(k)}+s)$, *respectivement. Utiliser la norme* ℓ_∞ *dans le calcul des régions de confiance.*

Étape 2 : Calcul d'un pas de progression. *Calculer le pas de progression* $s^{(k)}$ *avec l'algorithme* UVQCQP[1]. *Poser* $\tilde{x}^{(k)} = x^{(k)} + s^{(k)}$.

Étape 3 : Acceptation ou rejet du point-test. *Évaluer* $\Psi(\tilde{x}^{(k)},\sigma)$ *et le rapport*

$$\rho^{(k)} = \frac{\Psi(x^{(k)},\sigma) - \Psi(\tilde{x}^{(k)},\sigma)}{m^{(k)}(x^{(k)},\sigma) - m^{(k)}(\tilde{x}^{(k)},\sigma)} \tag{10.6}$$

Si $\rho^{(k)} \geq \eta_1$, *on définit* $x^{(k+1)} = \tilde{x}^{(k)}$ *; dans le cas contraire,* $x^{(k+1)} = x^{(k)}$.

Étape 4 : Mise à jour du rayon de confiance. *Poser*

$$\Delta^{(k+1)} = \begin{cases} \alpha_1 \|s^{(k)}\|_\infty & si\ \rho^{(k)} < \eta_1, \\ \Delta^{(k)} & si\ \eta_1 \leq \rho^{(k)} < \eta_2, \\ \alpha_2 \Delta^{(k)} & si\ \eta_2 \leq \rho^{(k)} \leq \eta_3, \\ \alpha_3 \Delta^{(k)} & si\ \rho^{(k)} > \eta_3 \end{cases} \tag{10.7}$$

Augmenter ensuite k d'une unité et retourner à l'étape 1.

Qu'entendons-nous exactement, à l'étape 2, par « approximation quadratique convexe » ? Il s'agit simplement d'une forme rendue convexe de l'approximation quadratique de Newton modifiée (3.34). Les fonctions $\check{\phi}^{(k)}(x)$ et $\hat{\phi}^{(k)}(x)$ correspondant à la fonction $\phi(x)$ sont, respectivement,

$$\check{\phi}^{(k)}(x^{(k)}+s) = \phi(x^{(k)}) + s^T \nabla_x \phi(x^{(k)}) + \frac{1}{2} s^T \check{H}^{(k)} s \tag{10.8}$$

$$\hat{\phi}^{(k)}(x^{(k)}+s) = -\phi(x^{(k)}) - s^T \nabla_x \phi(x^{(k)}) + \frac{1}{2} s^T \hat{H}^{(k)} s \tag{10.9}$$

où le Hessien de la fonction $\phi(x)$ au point $x^{(k)}$

$$\nabla_{xx}\phi(x^{(k)}) = Q^{(k)} \operatorname{diag}(\lambda_i)\, Q^{(k)T} \tag{10.10}$$

[1] Le sous-problème ainsi créé est bien de la forme (9.3).

est approché par

$$\check{H}^{(k)} = Q^{(k)}\,\mathrm{diag}\,[\max(\lambda_i, 0)]\; Q^{(k)T}, \tag{10.11}$$

$$\hat{H}^{(k)} = Q^{(k)}\,\mathrm{diag}\,[\max(-\lambda_i, 0)]\; Q^{(k)T}. \tag{10.12}$$

Notons que l'utilisation de la norme ℓ_∞ dans la définition de la région de confiance nous permet d'intégrer facilement les contraintes de bornes de (10.1) dans chaque sous-problème. Les contraintes de bornes envoyées à la routine UVQCQP sont en effet légèrement modifiées pour en tenir compte, elles s'expriment sous la forme

$$\max\left(\left[x_L - x^{(k)}\right]_i, -\Delta^{(k)}\right) \le [s]_i \le \min\left(\left[x_U - x^{(k)}\right]_i, \Delta^{(k)}\right). \tag{10.13}$$

La valeur de l'approximation locale (10.5) tout comme l'expression de ses dérivées directionnelles au point $x^{(k)}$ coïncident avec celles de $\Psi(x,\sigma)$ et les hypothèses 4.16 et 4.17 sont donc satisfaites. L'hypothèse 4.15 est également satisfaite puisque $m^{(k)}(x,\sigma)$ est une somme de fonctions continues au sens de Lipschitz et l'hypothèse 4.18 est sans objet : aucun paramètre ne varie d'une itération à l'autre. L'utilisation de l'algorithme UVQCQP pour la résolution du sous-problème garantit une décroissance suffisante (hypothèse 4.19) et le théorème 4.7 de convergence vers un point critique du premier ordre est dès lors d'application.

Notons que la valeur du paramètre σ n'est pas quelconque. Le théorème 8.2 exprime la valeur minimum que doit prendre σ pour que la fonction de mérite (10.2) soit exacte : la valeur de σ doit être supérieure ou égale à la norme ℓ_∞ des multiplicateurs de Lagrange relatifs aux contraintes du problème initial. Dans cette étude exploratoire, nous considérerons que σ a été choisi suffisamment grand par rapport au problème donné. Notons toutefois que, en pratique, la valeur de σ ne doit pas être trop importante pour ne pas dégrader les performances de l'algorithme.

10.2 Incompatibilité des contraintes.

Une des premiers difficultés lors de l'utilisation d'approximations locales quadratiques est la construction de sous-problèmes incompatibles, et ce même si le problème initial est convexe. Pour s'en convaincre, il suffit d'envisager le problème suivant.

Exemple 10.1 *Soit les deux contraintes convexes pour lesquelles il existe un espace admissible*

$$c_1(x,y) = x^4 + y^4 - 1 \le 0 \tag{10.14}$$

et

$$c_2(x,y) = (x-1)^2 + (y-1)^2 - 1 \le 0 \tag{10.15}$$

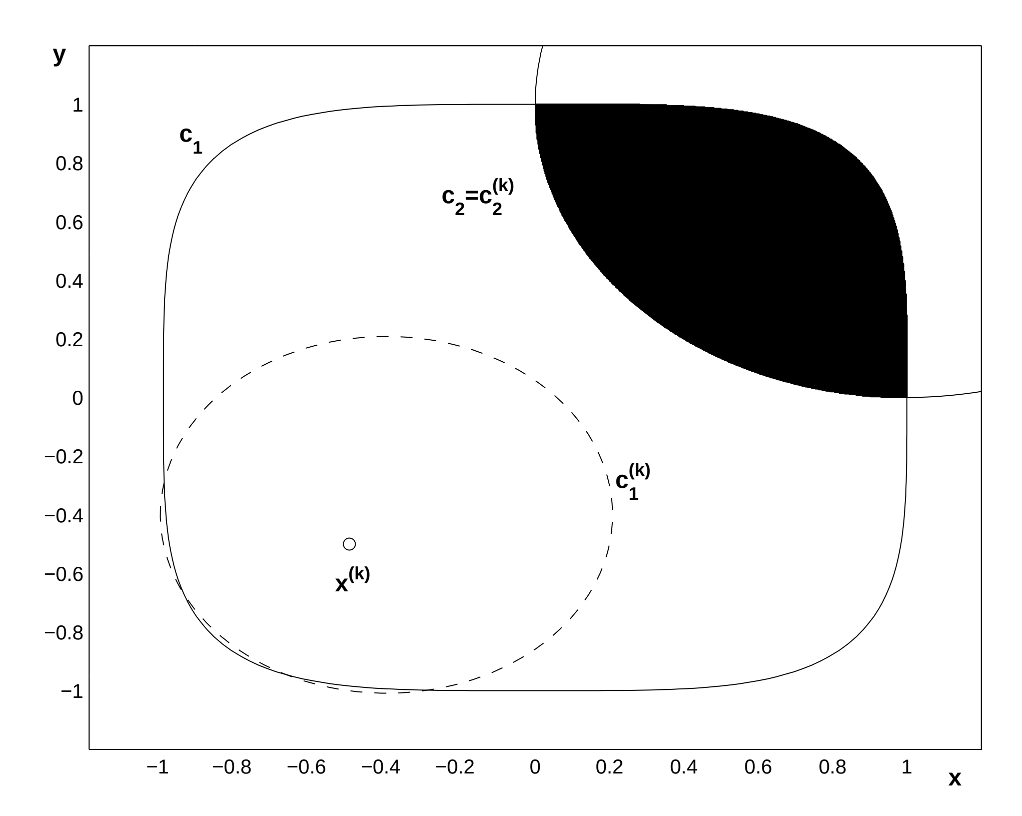


FIG. 10.1 – Illustration de l'exemple 10.1 au point $x^{(k)} = (-1/2, -1/2)$. Les deux contraintes $c_1(x,y)$ et $c_2(x,y)$ sont représentées par un trait continu et l'approximation locale de $c_1^{(k)}$ de c_1 par un trait discontinu. La contrainte quadratique c_2 est sa propre approximation locale. Nous pouvons constater que l'espace admissible pour les contraintes réelles (la zone noircie) disparaît si c_1 est remplacée par son approximation $c_1^{(k)}$.

au point $x^{(k)} = (-1/2, -1/2)$. *L'approximation locale quadratique pour la première contrainte est*

$$c_1^{(k)}(x,y) = -\frac{5}{8} + x + y + \frac{3}{2}\,x^2 + \frac{3}{2}\,y^2 \leq 0, \qquad (10.16)$$

alors que la seconde, étant déjà quadratique, est sa propre approximation. Dans le sous-problème ainsi créé, les deux contraintes approchées sont incompatibles (voir figure 10.1).

Notre approche n'a cependant aucune difficulté à contourner ce problème : le recours à une fonction de pénalité exacte tel qu'envisagé dans ce chapitre permettra de progresser vers le point minimisant une certaine pondération de la fonction objectif et de la violation des contraintes. L'exemple suivant illustre ceci.

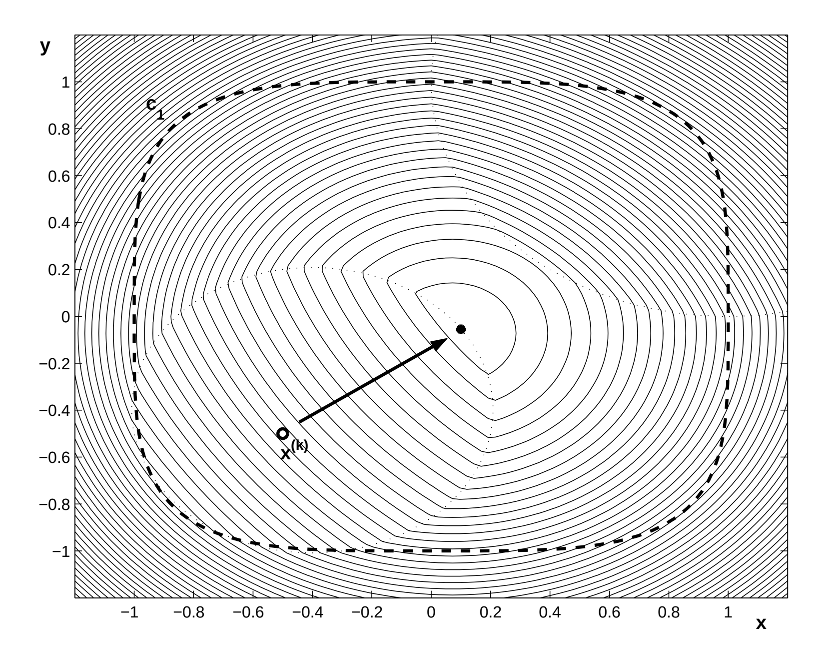


FIG. 10.2 – Illustration de l'exemple 10.2 : la figure montre l'approximation locale (10.18) construite pour le point $x^{(k)} = (-1/2, -1/2)$ avec $\sigma = 2$ de sorte que la fonction de pénalité est exacte. Au titre de point de repère, la contrainte réelle $c_1(x, y)$ est représentée en trait épais discontinu. Le minimum de ce problème ramène bien l'algorithme vers l'espace admissible du problème original.

Exemple 10.2 *Soit la fonction objectif linéaire*

$$f(x,y) = -x + y \tag{10.17}$$

soumise aux contraintes (10.14) *et* (10.15) *de l'exemple 10.1.*

Soit le point $x^{(k)} = (-1/2, -1/2)$ *comme itéré actuel. L'approximation locale de* $\Psi(x, y, \sigma)$ *est*

$$m^{(k)}(x,y,\sigma) = f(x,y) + \sigma \max[0, c_1^{(k)}(x,y)] + \sigma \max[0, c_2^{(k)}(x,y)]. \tag{10.18}$$

Cette fonction est représentée sur la figure 10.2 : le pas de progression engendré par cette représentation locale ramène sans difficulté l'algorithme vers la zone admissible si $\sigma > 1$. *La figure 10.3 montre les deux itérations suivantes.*

Une autre question importante dans le cadre des méthodes de type SQCQP est le sort réservé aux contraintes non-convexes. Ayant choisi de convertir chaque contrainte d'égalité en deux contraintes d'inégalité opposées, ce problème de non-convexité se pose forcément pour l'une des deux, sauf si la contrainte est linéaire. Pour comprendre la manière dont nous avons envisagé la gestion de contraintes non-convexes, nous pouvons considérer l'exemple suivant.

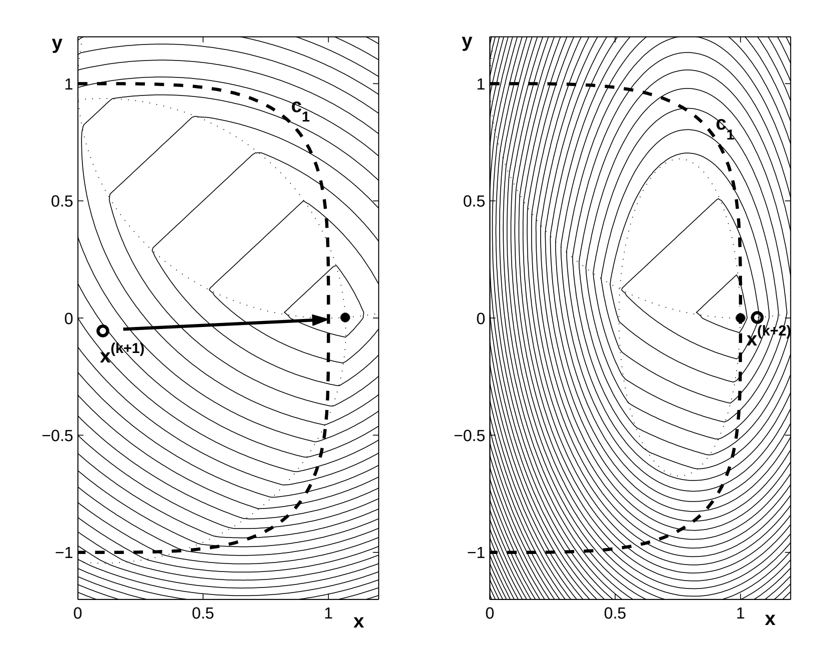


FIG. 10.3 – Illustration de l'exemple 10.2 : la figure montre l'approximation locale (10.18) construite pour les points $x^{(k+1)}$ (figure de gauche) et $x^{(k+2)}$ (figure de droite) avec $\sigma = 2$. Au titre de point de repère, la contrainte $c_1(x,y)$ est représentée en trait épais discontinu.

Exemple 10.3 *Soit la fonction objectif linéaire*

$$f(x,y) = -x - y \tag{10.19}$$

soumise à la contrainte d'égalité non-convexe

$$c(x,y) = x^3 + y^2 - 1 = 0. \tag{10.20}$$

Au point $x^{(k)} = (1/4, 5/4)$*, les deux approximations locales quadratiques correspondantes sont*

$$c^+(x,y) = -\frac{63}{64} - \frac{3}{16}x + \frac{3}{4}x^2 + y^2 \leq 0, \tag{10.21}$$

$$c^-(x,y) = -\frac{83}{32} - \frac{3}{16}x - \frac{5}{2}y \leq 0. \tag{10.22}$$

Ces deux contraintes sont incompatibles (voir figure 10.4). L'approximation locale de $\Psi(x,y,\sigma)$ *est*

$$m^{(k)}(x,y,\sigma) = f(x,y) + \sigma \max[0, c^+(x,y)] + \sigma \max[0, c^-(x,y)]. \tag{10.23}$$

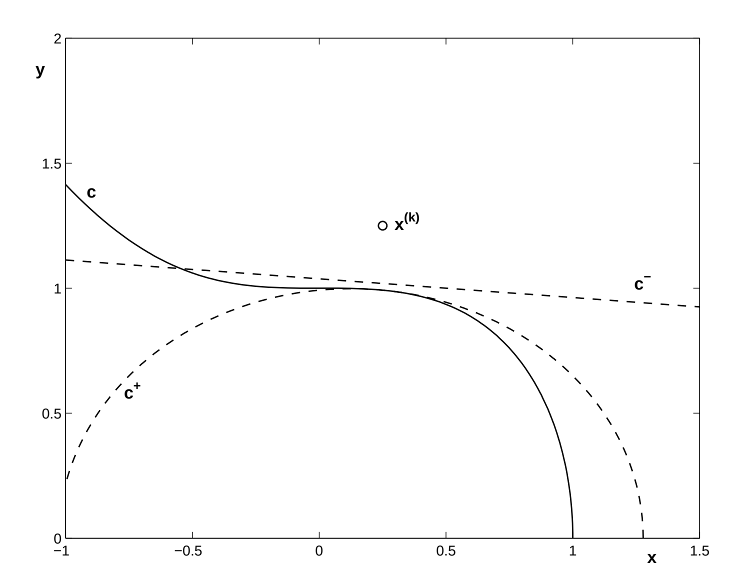


FIG. 10.4 – Illustration de l'exemple 10.3 au point $x^{(k)} = (1/4, 5/4)$. La contrainte $c(x,y)$ est représentée par un trait continu et les approximations locales de c^+ et c^- par des traits discontinus.

Cette fonction est représentée sur la figure 10.5 : le pas de progression engendré par cette représentation locale ramène sans difficulté l'algorithme vers la contrainte d'égalité. La figure 10.6 montre l'itération suivante.

10.3 Performances sur un petit ensemble de CUTEr.

Pour évaluer les potentialités d'un algorithme construit autour de la « brique élémentaire » UVQCQP, nous avons sélectionné quelques problèmes de petite taille de l'ensemble de problèmes de test CUTEr (voir Bongartz *et al.* [5] et Gould *et al.* [47]). Les trente-six problèmes sélectionnés (voir tableau 10.1) sont tous ceux répondant aux critères suivants : il sont non-linéaires, ont une dimension $n \leq 30$, un nombre de contraintes $|\mathcal{E}| + |\mathcal{I}| \leq 30$ et les dérivées premières et secondes sont disponibles.

Le nombre d'itérations[2] et la valeur σ choisie pour chaque problème sont présentés dans le tableau 10.1. Les résultats de trois autres algorithmes bien connus (KNITRO, LOQO et IPOPT sont des méthodes de type *point intérieur*) sont également portés dans ce tableau, ceux-ci proviennent des travaux de Wächter et Bie-

[2] Pour l'algorithme SQCQP, le nombre d'itérations correspond au nombre d'évaluation de la fonction objectif et des contraintes.

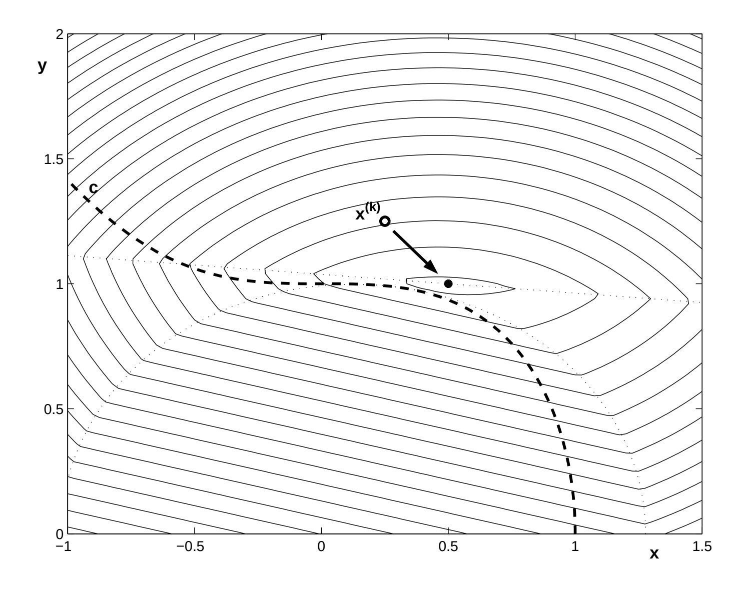


FIG. 10.5 – Illustration de l'exemple 10.3 : la figure montre l'approximation locale (10.23) construite pour le point $x^{(k)} = (-1/2, -1/2)$ avec $\sigma = 2$. Au titre de point de repère, la contrainte $c(x, y)$ est représentée en trait épais discontinu. Le minimum de ce problème ramène bien l'algorithme vers la contrainte d'égalité du problème original.

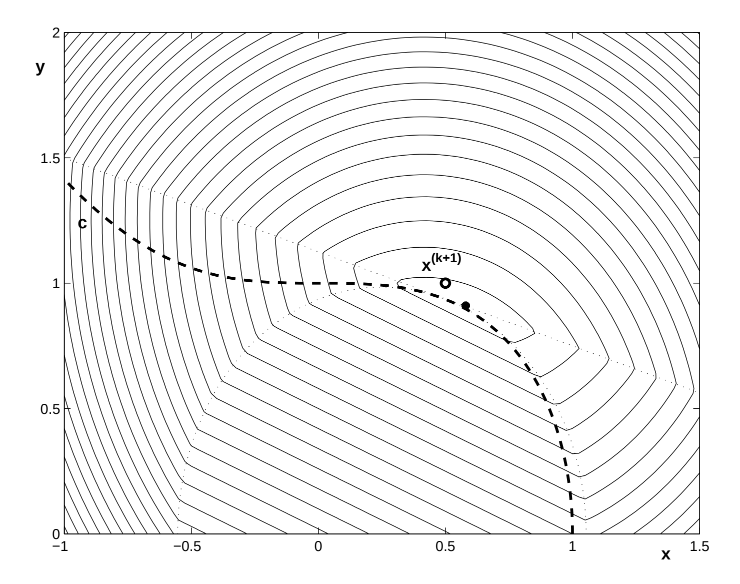


FIG. 10.6 – Illustration de l'exemple 10.3 : la figure montre l'approximation locale (10.23) construite pour le point $x^{(k+1)}$ avec $\sigma = 2$. Au titre de point de repère, la contrainte $c(x, y)$ est représentée en trait épais discontinu.

TAB. 10.1 – Ensemble des 36 problèmes de test CUTEr sélectionnés. Pour chaque problème, sont donné les nombres d'évaluations de fonction des algorithmes KNITRO, LOQO, IPOPT et SQCQP. Le tableau donne aussi la valeur de σ utilisée dans la fonction de mérite (10.2). La présence d'un astérisques signifient que l'algorithme a convergé vers un point non admissible et deux astérisque signifie que le maximum autorisé d'itérations a été atteint avant la convergence.

Nom	n	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{I}\rvert$	KNITRO	LOQO	IPOPT	SQCQP	σ
BT11	5	1	2	8	12	9	11	2
BT6	5	2	0	12	13	18	9	5
CRESC4	6	0	8	55	483	23175*	11	10^5
DIPIGRI	7	0	4	9	26	22	7	10
HS100	7	0	4	9	26	21	7	10
HS100LNP	7	2	0	9	12	21	7	$1,2$
HS100MOD	7	0	4	14	28	29	7	$1,2$
HS101	7	0	5	683	3680	130	87	10^4
HS102	7	0	5	3001*	155	25	39	10^4
HS103	7	0	5	3002*	91	34	43	10^4
HS104	8	0	5	16	15	9	11	10^4
HS107	9	6	0	7	16	11	68	2.10^4
HS109	9	6	4	3001*	52	2247	32	10^3
HS111	10	3	0	11	16	16	4	10^5
HS111LNP	10	3	0	11	18	16	4	10^5
HS40	4	3	0	4	9	4	3	10^4
HS46	5	2	0	22	30	20	60	$0,1$
HS47	5	3	0	18	55	21	9	$0,1$
HS56	7	4	0	37	21	11	9	2
HS68	4	2	0	19	48	24	189	10
HS69	4	2	0	13	17	13	57	50
HS71	4	1	1	8	14	9	15	1
HS74	4	3	2	13	15	11	73	6
HS75	4	3	2	12	17	12	145**	10
HS77	5	2	0	12	13	13	20	$0,1$
HS78	5	3	0	5	8	5	4	10^4
HS79	5	3	0	5	8	5	6	$0,1$
HS80	5	3	0	10	10	8	4	10^3
HS81	5	3	0	10	17	8	4	10^3
HS93	6	0	2	7	13	10	5	10^5
HS99	7	2	0	4	18	7	5	10^6
LAUNCH	25	9	19	39	76	27	5	100
SYNTHES1	6	0	6	8	17	10	5	10
SYNTHES2	11	1	13	13	22	26	3	50
SYNTHES3	17	2	21	11	21	18	3	50
TWOBARS	2	0	2	8	11	10	7	10

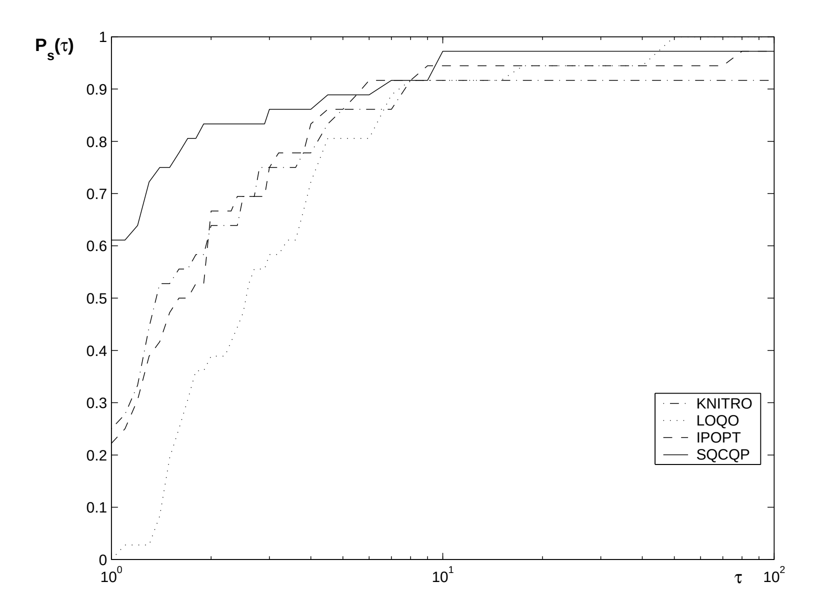


FIG. 10.7 – Profils de performances correspondant au tableau 10.1. La figure 10.7 présente les profils de performance correspondant au tableau 10.1. Les résultats semblent prometteurs : plus de 60% des problèmes sont résolus plus rapidement par notre algorithme SQCQP et seul LOQO s'avère finalement plus robuste.

gler [102]. La comparaison porte sur le nombre d'évaluation de la fonction objectif et des contraintes mais les résultats obtenus doivent être nuancés car les critères d'arrêt des différents algorithmes sont différents et la comparaison ne peut donc être envisagée qu'en terme d'ordre de grandeur.

La figure 10.7 présente les profils de performance correspondant au résultats du tableau 10.1. Les premiers résultats semblent prometteurs : plus de 60% des problèmes sont résolus plus rapidement par SQCQP et il faut attendre $\tau = 50$ avant de voir la courbe IPOPT repasser momentanément au dessus à deux reprises. *In fine*, seul l'algorithme LOQO peut se prévaloir d'être plus robuste que SQCQP. Globalement, il apparaît clairement sur le figure 10.7 que notre ébauche d'algorithme SQCQP tient la comparaison et « domine » les autres méthodes pour peu que le paramètre σ soit initialisé correctement. Or, celui-ci a été calibré de manière *ad hoc* et les résultats sont donc quelque biaisés. Il est clair que l'estimation initiale de σ et sa mise à jour au cours du calcul seront des facteurs-clés de la mise au point d'un algorithme complet.

10.4 Conclusion

L'objet de ce chapitre est de présenter le potentiel de la « brique élémentaire » UVQCQP décrite au chapitre 9. Un algorithme sommaire est donc présenté. Celui-ci n'a qu'une vocation exploratoire : de nombreuses questions restent en suspens avant de pouvoir utiliser un algorithme de ce type. Comment calibrer le paramètre de pénalité σ ? La stratégie de découplage et de convexification des contraintes est-elle la plus efficace ? Comment estimer les multiplicateurs de Lagrange ? Etc.

Via quelques exemples, nous avons tenter de faire comprendre les avantages que pourraient receler l'utilisation de la routine UVQCQP. Trois forces peuvent ainsi être mises en évidence.

- L'éventuelle incompatibilité des contraintes quadratiques est gérée de façon très simple et naturelle. Il n'est donc pas nécessaire d'élaborer de stratégie de relaxation des contraintes.
- Les régions de confiance de type ℓ_∞ sont aisément prises en compte, tout comme les éventuelles contraintes de bornes. La stratégie d'activation et de désactivation des contraintes de bornes est d'ailleurs gérée au niveau des sous-problèmes eux-mêmes grâce à l'introduction de l'espace $\mathcal{W}^{(k)}$ au chapitre 9.
- Les sous-problèmes sont résolus de manière exacte.

Les quelques expériences numériques effectuées sur des problèmes de petites tailles issus de l'ensemble de test CUTEr s'avèrent encourageants. Les performances de notre approche SQCQP semblent en effet pouvoir rivaliser avec des algorithmes bien connus dans la littérature.

Quatrième partie

Conclusion et perspectives

Chapitre 11

Conclusion et perspectives

Ce chapitre est destiné à tirer les conclusions de ce travail et à dresser les nombreuses perspectives encore à explorer. Nous avons présenté, dans le chapitre 2, les grandes familles de méthodes de globalisation et, dans le chapitre 3 différentes possibilités d'approximations locales pour un problème d'optimisation. Le chapitre 4 présente le cadre théorique nécessaire à la bonne compréhension des différents concepts liés à la convergence des méthodes par régions de confiance. Sur cette base, nous nous sommes attelés à utiliser ce formalisme pour explorer en profondeur l'utilisation d'approximations locales quadratiques dans des problèmes contraints et non-contraints.

Les choix que nous avons opérés ne sont, bien entendu, pas les seuls possibles : d'autres approximations locales, combinées avec d'autres formes de régions de confiance mèneront encore probablement à de nouveaux algorithmes dont les propriétés seront éclairées par le formalisme théorique du chapitre 4.

Les méthodes d'optimisation convexe, développées dans le domaine de l'optimisation des structures (*e.g.* Fleury et Braibant [36], Svanberg [96, 97], Bruyneel [9]), pourraient par exemple être utilement exprimées sous le formalisme des régions de confiance. En effet, les *move limits* généralement utilisées dans ce cadre ne sont finalement rien d'autre qu'une région de confiance ℓ_∞. Outre le bénéfice théorique d'assurer une convergence globale, ce formalisme ouvre la porte à l'utilisation d'approximations locales non-convexes.

Mais les apports majeurs de ce travail commencent au chapitre 5.

11.1 Calibration de modèles mathématiques.

L'identification paramétrique est une étape clé dans le développement d'un modèle mathématique. Les modèles développés, quelles que soient les disciplines, sont de plus en plus complexes. Les paramètres sont dès lors de plus en plus

nombreux et leur calibration de plus en plus difficile.

S'il est parfois possible, en physique, d'évaluer la valeur d'un paramètre tel que la conductivité, la masse volumique ou la chaleur massique sur base des propriétés microscopiques, il n'en est pas de même pour les domaines de modélisation du vivant, des sciences économiques ou des sciences humaines. L'étape de calibration du modèle est dès lors indispensable. En fonction des caractéristiques de son problème, le modélisateur trouvera dans le chapitre 5 un bon résumé des questions à se poser *après* la modélisation.

La question qui se pose alors naturellement est celle de l'observabilité de ces paramètres. Les relevés expérimentaux dont nous disposons sont-ils suffisants pour déduire les valeurs des paramètres de notre modèle ? L'*expérience jumelle* sert à répondre à cette question.

L'identification des paramètres peut être mise sous la forme d'un problème d'optimisation généralement non-contraint ou quasi non-contraint (n'ayant que des contraintes de bornes) et résolu par les différentes méthodes abordées dans ce travail. La fonction objectif à établir mesure l'écart entre le modèle numérique et les relevés d'expérience tandis que les variables à optimiser sont les paramètres à calibrer.

L'augmentation de la complexité des modèles mathématiques n'est toutefois pas sans conséquence sur l'optimisation à opérer pour l'identification des paramètres car une plus grande complexité se traduit généralement par un temps de calcul plus important et une utilisation intensive des ressources informatiques. Or les méthodes d'optimisation sont itératives et le nombre de simulations nécessaires au calibrage peut s'avérer relativement important. C'est pourquoi, tout au long de ce travail, nous nous sommes attelés à réduire le nombre d'itérations nécessaires.

Il convenait donc d'utiliser les meilleures méthodes d'optimisation disponibles et celles-ci font invariablement appel aux dérivées de la fonction objectif par rapport aux paramètres à calibrer. Celles-ci peuvent être obtenues par le biais de différentes méthodes : les *différences finies*, la *différentiation directe* ou le *modèle adjoint*. Chaque méthode a ses avantages et inconvénients qui sont synthétisés dans le tableau 5.1.

11.2 Trust, un algorithme fiable.

Dans le cadre de ce travail, nous avons développé une routine FORTRAN nommée Trust. Celle-ci est présentée et testée aux chapitre 6 et 7. Il s'agit d'un algorithme utilisant une globalisation par régions de confiance et des approximations globales quadratiques pour la fonction objectif.

Plusieurs variantes sont étudiées et comparées. L'algorithme se base sur le

schéma général décrit au chapitre 4. Il utilise des approximations quadratiques pour la fonction objectif et prend en compte des contraintes de bornes. Plusieurs versions ont été développées utilisant des approximations locales de type quasi-Newton ainsi qu'une version utilisant l'approximation locale de Gauss–Newton. La routine développée inclut également des fonctionnalités pour faciliter la mise à échelle et l'impression personnalisée.

Une identification paramétrique est sommairement développée. Il s'agit d'un système dynamique simple, à titre de d'exemple : le modèle de Lotka–Volterra. Les performances de différentes versions de Trust sont analysées et comparées à la routine M1QN3 de Gilbert et Lemaréchal [40].

Afin de fournir une analyse plus détaillée du comportement de Trust pour ce problème, une analyse de la robustesse des différentes méthodes a été effectuée en variant les points de départ. La globalisation par régions de confiance apparaît, dans ce cas test, plus efficace que l'approche par recherche linéaire. De plus, les deux versions dites « inconditionnelles » qui utilisent l'information issue des itérations infructueuses sont plus efficaces que leurs homologues conditionnelles. L'approche utilisant une mise à jour BFGS inconditionnelle de l'approximation de la matrice hessienne s'avère être la plus performante, aussi bien du point de vue de la robustesse que de la vitesse de convergence.

11.3 Les itérations trop réussies.

Le chapitre 7 établit une nouvelle proposition de mise à jour du rayon de la région confiance dans les algorithmes du même nom. Traditionnellement, un rapport $\rho^{(k)}$ fournit une mesure de la fidélité de l'approximation locale par rapport à la véritable fonction objectif dans le voisinage de l'itéré courant. Cette mesure est ensuite utilisée pour mettre à jour le rayon de confiance d'une itération à l'autre. Les règles empiriques habituelles opèrent une réduction du rayon de confiance après une itération infructueuse et le maintiennent constant ou l'accroissent après une itération réussie.

Notre travail établit que la stratégie de mise à jour du rayon de confiance est susceptible d'avoir une forte influence sur les performances de l'algorithme. Ce sujet — critique du point de vue de l'efficacité de l'algorithme — avait été relativement peu traité jusqu'à présent.

Dans l'approche générale des régions de confiance, certaines itérations sont appelées les itération *très réussies* parce qu'elles fournissent des diminutions importantes de la fonction objectif. L'approche habituelle dans un tel cas est d'élargir la région de confiance. La raison sous-jacente à cette augmentation est intuitive : l'approximation locale semble précise dans une grande région autour de l'itéré courant et l'algorithme devrait dès lors être autorisé à effectuer un pas plus grand

si nécessaire.

Le propre de notre travail est d'introduire une nouvelle catégorie d'itérations. Pensant que les itérations très réussies galvaudent leur nom si la réduction obtenue pour la fonction objectif est bien trop importante, nous introduisons le concept des itérations *trop* réussies. Dans ce cas, le rayon de la région de confiance est maintenu quasi-constant. Bien entendu, cette stratégie modifiée satisfait aux hypothèses du chapitre 4 et les propriétés générales de convergence globale sont encore d'application.

De manière générale, cette nouvelle technique est substantiellement plus performante tant du point de vue de l'efficacité que du point de vue de la robustesse. Cette plus grande efficacité est clairement démontrée par les tracés des profils de performance sur un ensemble de test. Cette stratégie est très intuitive et largement applicable. Il est intéressant de noter que, lorsque l'algorithme est proche de la convergence, la plupart des itérations sont très réussies et le taux de convergence n'est donc pas affecté.

Notons également que les expériences numériques utilisent des approximations locales quadratiques de type quasi-Newton. Elles montrent clairement que la nouvelle approche améliore la vitesse de l'algorithme. En effet, cette règle permet d'éviter la pollution de l'approximation du Hessien avec des mises à jour de type quasi-Newton imprécises. Malgré une légère détérioration de la robustesse, il apparaît que l'algorithme le plus efficace combine cette nouvelle stratégie avec une mise à jour inconditionnelle de l'approximation de la matrice hessienne par une règle de type quasi-Newton.

Il reste désormais à valider ce concept très simple avec d'autres types d'approximations locales (linéaires, coniques, asymptotes mobiles, ...), avec des problèmes contraints et avec des problèmes de plus grande taille.

11.4 De l'utilisation d'une approche SQCQP.

Le chapitre 9 présente un algorithme permettant de résoudre un type de sous-problèmes engendrés par une approche séquentielle de problèmes quadratiques à contraintes quadratiques. Il s'agit d'une « brique élémentaire » pour qui s'attaque à une fonction de pénalité de type ℓ_1 avec des approximations quadratiques convexes. Le sous-problème engendré est donc convexe, non-différentiable, quadratique par morceaux et soumis à des contraintes de bornes.

L'originalité de l'algorithme UVQCQP réside dans l'utilisation d'un développement théorique proposé par Lemaréchal *et al.* [65]. L'espace d'optimisation est divisé en trois sous-espaces : celui des contraintes de bornes actives $\mathcal{W}$ d'une part et son complément orthogonal qui est lui-même décomposé en deux sous-espaces $\mathcal{U}$ et $\mathcal{V}$. L'espace $\mathcal{U}(x)$ est le plus grand sous-espace dans lequel la fonction est

différentiable au voisinage de x.

Si la sous-routine UVQCQP s'avère plutôt performante, il reste à construire un algorithme global qui l'utilisera au maximum de ses possibilités. Le chapitre 10 en brosse une esquisse mais celle-ci — bien qu'encourageante — est insuffisante. De nombreuses questions restent en suspens et constituent autant de perspectives intéressantes. Nous pouvons citer, par exemple.

– La calibration du paramètre de pénalité σ qu'il faut choisir suffisamment grand pour assurer que le minimum de la fonction de pénalité corresponde bien à la solution du problème mais qu'il faut se garder de surdimensionner pour ne pas altérer les performances de l'algorithme. Il est probable que la réponse à cette question cruciale passe par une adaptation du paramètre σ au fur et à mesure des itérations.
– La gestion des contraintes d'égalité peut certainement être améliorée. Il n'est pas certain que la décomposition en deux inégalités opposées soit l'approche la plus performante : nous pourrions, par exemple, envisager l'utilisation de variables d'écart.
– La « convexification » de la fonction objectif ou des contraintes est également une source d'intérêt : nous proposons de « mettre à zéro » les courbures négatives mais ce choix est discutable.

L'approche UVQCQP privilégie des approximations locales quadratiques tant pour la fonction objectif que pour les contraintes. Dans les problèmes réels, il est cependant relativement rare de disposer des dérivées secondes de ces fonctions et les approximations locales doivent dès lors se construire sur base d'informations glanées lors des itérations précédentes : les plus connues de ces stratégies étant les mise à jour de type quasi-Newton. Toutes les questions de mise à jour conditionnelle ou inconditionnelle que nous nous sommes posées dans le chapitre 7 doivent donc également être reposées dans ce cadre.

Notons aussi que l'algorithme UVQCQP fait un grand usage de matrices : au moins une matrice hessienne (de dimension $n \times n$) pour chacune des m contraintes. Ceci rend la méthode inutilisable pour des problèmes de grande taille et/ou avec un grand nombre de contraintes. À l'image de ce qui s'est fait pour d'autres méthodes, il serait intéressant d'élaborer une version approchée de l'algorithme initial capable de gérer ce genre de problèmes. Nous pensons, par exemple, à l'utilisation d'approximations quadratiques séparables qui ont déjà montré leur efficacité dans ce type de problèmes.

Malgré ses performances satisfaisantes, il reste une ombre au tableau de la méthode UVQCQP : c'est sa limitation au cas convexe. Sans cette limitation, point besoin de « convexifier » les contraintes au niveau de SQCQP et ceci constituerait un avantage indéniable dans la prise en compte des contraintes d'égalité. Or, la clef de voûte de la méthode UVQCQP, à savoir la décomposition $\mathcal{U}\mathcal{V}$, a initialement été développé pour des problèmes convexes. Avant même de pouvoir modifier

l'algorithme, il faudrait étendre la base théorique de la théorie sous-jacente.

Cinquième partie

Annexes

Annexe A

Routines FORTRAN : mode d'emploi

Les sous-routines d'optimisation proprement dites ont été implémentées dans des *modules* FORTRAN90. Les routines d'optimisation se nomment

- `trust_SR1_c` pour Trust-SR1 conditionnelle,
- `trust_BFGS_c` pour Trust-BFGS conditionnelle,
- `trust_SR1_i` pour Trust-SR1 inconditionnelle,
- `trust_BFGS_i` pour Trust-BFGS inconditionnelle
- et `trust_GN` pour Trust-GN.

Pour les quatre premières routines, les séquences d'appel sont identiques

```
call trust_SR1_c(simul,n,x,f,g,delta,deltamax,epsg,impres,&
&    io,mode,succes,niter,lbnd,ubnd,freex,valcar,varcar,hes)
```

et pour la dernière

```
call trust_GN(simulGN,n,x,m,c,jacob,delta,deltamax,epsg,&
&    impres,io,mode,succes,niter,lbnd,ubnd,freex,valcar,&
&    varcar,hesGN).
```

A.1 Le simulateur.

Le premier argument de chaque routine est un « simulateur », une routine FORTRAN fournie par l'utilisateur et qui servira à calculer les valeurs de la fonction objectif, du gradient ou de la matrice jacobienne. Elle se présente comme suit :

- `simul(indic,n,x,f,g)` ou
- `simulGN(indic,n,m,x,c,cref,jacob)` pour la version Gauss-Newton.

Les significations des différents arguments de ces routines sont énumérées ci-dessous.

A.1.1 Indicateur.

```
integer,intent(inout) :: indic
```

La variable `indic` est un entier qui établit la communication entre le simulateur et la routine d'optimisation.

en entrée : l'utilisateur doit faire en sorte que la routine `simul` (resp. `simulGN`)
- ne fasse aucune opération (hormis des impressions) si `indic` $= 1$;
- calcule `f` et `g` (resp. `c` et `jacob`) si `indic` $= 2$.

en sortie : l'utilisateur doit faire en sorte que la routine `simul` (resp. `simulGN`) renvoie pour `indic`
- une valeur strictement positive si le simulateur n'a rencontré aucun problème particulier ;
- une valeur nulle si le simulateur demande l'arrêt de l'optimisation (par exemple parce que la fonction objectif a atteint une valeur cible) ;
- une valeur négative si le simulateur est incapable de calculer les valeurs ou de faire les opérations qui lui sont demandées.

A.1.2 Les entrées.

```
integer,intent(in) :: n
real(kind=8),dimension(n),intent(in) :: x
integer,intent(in) :: m
real(kind=8),dimension(m),intent(in) :: cref
```

La variable `n` est un entier, elle donne la dimension n du vecteur des variables à optimiser x. La variable `x` donne au simulateur la valeur des variables pour lesquelles la routine d'optimisation demande à calculer la valeur de la fonction objectif et de sa dérivée.

Le cas échéant, la variable `m` est un entier qui donne la dimension m du vecteur $c(x) - \hat{c}$ que l'utilisateur souhaite minimiser au sens des moindres carrés. La variable `cref`, quant à elle, est le vecteur des valeurs de référence $\hat{c}$.

Toutes ces variables doivent demeurer inchangées en sortie.

A.1.3 Les sorties.

```
real(kind=8),intent(inout) :: f
real(kind=8),dimension(n),intent(inout) :: g
real(kind=8),dimension(m),intent(inout) :: c
real(kind=8),dimension(n,m),intent(inout) :: jacob
```

Si l'appel du simulateur se fait avec `indic` $= 1$, la routine d'optimisation fait en sorte que les variables `f` et `g` (resp. `c` et `jacob`) contiennent, *en entrée*, les valeurs de la fonction objectif $f(x)$ et de son gradient $g(x)$ (resp. $c(x)$ et $G(x)$) au point `x`. Cette disposition permet à l'utilisateur de faire imprimer ces valeurs et d'ainsi formater les impressions de la routine d'optimisation comme il le souhaite. Dans ce cas, les valeurs de ces variables doivent rester inchangées en sortie.

Si l'appel de la routine se fait avec `indic` $= 2$, l'utilisateur doit faire en sorte que les variables `f` et `g` (resp. `c` et `jacob`) contiennent, *en sortie*, les valeurs de la fonction objectif $f(x)$ et de son gradient $g(x)$ (resp. $c(x)$ et $G(x)$) au point `x`. Dans ce cas, aucune valeur ne doit être spécifiée en entrée.

A.2 Les autres variables.

```
integer,intent(in) :: n
```

La variable `n` est un entier ; elle donne la dimension n du vecteur des variables à optimiser x. Cette variable demeure inchangée en sortie.

```
real(kind=8),intent(inout),dimension(n) :: x
```

En *entrée*, la variable `x` doit être la valeur de départ $x^{(0)}$ des variables à optimiser. En *sortie*, cette variable contient les valeurs optimisées.

```
real(kind=8),intent(inout) :: f
```

En *entrée*, la variable `f` doit être la valeur de départ $f(x^{(0)})$ de la fonction objectif à optimiser. En *sortie*, cette variable contient la valeur finale de la fonction objectif à optimiser.

```
real(kind=8),intent(inout),dimension(n) :: g
```

En *entrée*, la variable `g` doit être la valeur de départ $g(x^{(0)})$ du gradient de la fonction objectif. En *sortie*, cette variable contient la valeur finale du gradient la fonction objectif.

```
integer,intent(in) :: m
```

La variable `m` est un entier ; elle donne la dimension m du vecteur $c(x) - \hat{c}$ à minimiser au sens des moindres carrés. Cette variable demeure inchangée en sortie.

```
real(kind=8),dimension(m),intent(inout) :: c
```

En *entrée*, la variable `c` doit être le vecteur des valeurs de départ $c(x^{(0)})$. En *sortie*, cette variable contient les valeurs du vecteur $c(x)$ lors de la dernière itération.

```
real(kind=8),intent(inout),dimension(n,m) :: jacob
```

En *entrée*, la variable `jacob` doit être la valeur de départ $G(x^{(0)})$ de la matrice jacobienne. En *sortie*, cette variable contient la valeur de la matrice jacobienne $G(x)$ lors de la dernière itération.

```
real(kind=8),intent(inout) :: delta
```

En *entrée*, la variable `delta` doit être la valeur de départ du rayon de confiance $\Delta^{(0)}$. En *sortie*, cette variable contient la valeur finale du rayon de confiance.

```
real(kind=8),intent(inout) :: deltamax
```

En *entrée*, la variable `delta` doit être la valeur maximum $\Delta_{\max}$ du rayon de confiance. Cette variable est inchangée en sortie.

```
real(kind=8),intent(inout) :: epsg
```

En *entrée*, la variable `epsg` doit être plus petite que l'unité, c'est la valeur du paramètre ε du critère d'arrêt (6.37). En *sortie*, cette variable contient la valeur finale effective de $\varepsilon^{(k)}$ défini par (6.35).

```
integer,intent(in) :: impres
```

En *entrée*, la variable `impres` contrôle les impressions de la routine, sa valeur est inchangée en sortie. Les différentes options d'impression sont :

- `impres` < 0 : l'optimisation se fait silencieusement. Cependant, toutes les $-$`impres` itération(s), un appel du simulateur `simul` est effectué pour lequel `indic=1`. Ceci permet à l'utilisateur de formater les sorties à sa meilleure convenance.
- `impres` $= 0$: aucune impression, l'optimisation se fait silencieusement.
- `impres` $= 1$: messages d'erreur, impression finale et initiale uniquement.
- `impres` $= 2$: une impression par itération dont la valeur de la fonction objectif.
- `impres` ≥ 3 : commentaires détaillés sur le comportement de l'algorithme.

```
integer,intent(in) :: io
```

La variable `io` désigne l'unité dans laquelle les sorties sont écrites. Si `io` $= 6$, l'impression se fait directement à l'écran. Cette variable est inchangée en sortie.

```
integer,intent(out) :: mode
```

La variable `mode` est une variable de sortie. Elle indique le mode d'arrêt de la routine d'optimisation :

- `mode` < 0 : le simulateur `simul` a fourni une valeur de sortie `indic<0`.
- `mode` $= 0$: le simulateur a demandé l'arrêt en retournant la valeur `indic` $=$ 0.
- `mode` $= 1$: Arrêt normal. Le critère d'arrêt de décroissance du gradient est atteint.
- `mode` $= 2$: un des arguments d'entrée n'est pas bien initialisé.
- `mode` $= 3$: le nombre maximum d'itérations a été atteint.

```
integer, intent(out) :: succes
```

La variable `mode` est une variable de sortie. Elle indique le nombre d'itérations réussies au cours de l'optimisation.

```
integer,intent(inout) :: niter
```

En *entrée*, la variable `niter` indique à l'algorithme le nombre maximum d'itérations qu'il peut effectuer. En *sortie*, cette variable donne le nombre effectif d'itérations effectuées.

```
real(kind=8),intent(in),dimension(n) :: lbnd
real(kind=8),intent(in),dimension(n) :: ubnd
```

Les variables `lbnd` et `ubnd` désignent, respectivement, les bornes inférieures $\underline{x}$ et supérieures $\overline{x}$ imposées aux variables. Ces variables sont inchangées en sortie.

```
integer,intent(in),dimension(n),optional :: freex
```

La variable optionnelle `freex` permet de fixer une ou plusieurs variables. La variable `i` est libre si `freex(i)` $= 0$ et fixée si `freex(i)` $= 1$. Cette variable est inchangée en sortie. En l'absence de `freex`, toutes les variables sont libres par défaut.

```
real(kind=8),intent(in),dimension(n),optional :: valcar
real(kind=8),intent(in),dimension(n),optional :: varcar
```

Les variables `valcar` et `varcar` désignent, respectivement, les valeurs caractéristiques x_i^{car} et les variations caractéristiques $\delta x_i^{\mathrm{car}}$ utilisées dans (6.29) pour mettre le problème à échelle. Ces variables sont inchangées en sortie. En l'absence de `valcar`, les valeurs caractéristiques sont nulles par défaut et, en l'absence de `varcar`, les variations caractéristiques sont unitaires par défaut.

```
real(kind=8),intent(inout),dimension(n,n),optional :: hes
```

En *entrée*, la variable optionnelle `hes` permet de spécifier une approximation initiale $H^{(0)}$ de la matrice hessienne. En *sortie*, la variable `hes` contient l'approximation $H^{(k)}$ de la matrice hessienne à la fin du processus d'optimisation.

```
real(kind=8),intent(out),dimension(n,n),optional :: hesGN
```

En *sortie*, la variable `hesGN` contient l'approximation $H^{(k)}$ de la matrice hessienne à la fin du processus d'optimisation.

A.3 Exemple d'utilisation

Voici un exemple d'utilisation de la routine `trust_BFGS_i`. Trust se présente sous la forme d'un *module* FORTRAN95 nommé `trust_BFGS_i_m`, la routine devient donc accessible depuis le programme principal grâce à l'instruction

`'use trust_BFGS_i_m'.`

Il s'agit du problème de Rosenbrock logarithmique (7.14).

A.3.1 Programme principal

```
program main

  use simul_m
  use trust_BFGS_i_m

   (déclaration des variables)
   (initialisation des variables)

  indic=2
  call simul(indic,n,x,f,g)
  if (indic.le.0) stop

  open(unit=io,file='output.txt')
  call trust_BFGS_i(simul,n,x,f,g,delta,deltamax,epsg,impres,&
       &io,mode,succes,niter,lbnd,ubnd,freex,valcar,varcar,hes)
  close(io)

end program main
```

A.3.2 Simulateur

```
module simul_m
contains
  subroutine simul(indic,n,x,f,g)
    integer,intent(inout) :: indic
    integer,intent(in) :: n
    real(kind=8),dimension(n),intent(in) :: x
    real(kind=8),intent(inout) :: f
    real(kind=8),dimension(n),intent(inout) :: g
    integer :: i,j

    if (indiq.eq.2) then
       f=log(1+10000*(x(2)-x(1)**2)**2+(1-x(1))**2)
       g(1)=(-40000*(x(2)-x(1)**2)*x(1)-2*(1-x(1)))&
            &/(1+10000*(x(2)-x(1)**2)**2+(1-x(1))**2)
       g(2)=20000*(x(2)-x(1)**2)&
            &/(1+10000*(x(2)-x(1)**2)**2+(1-x(1))**2)
    end if

  end subroutine simul
end module simul_m
```

Annexe B

Zéros des polynômes du troisième et du quatrième degré

Cette annexe présente les formules exactes utilisées pour calculer les zéros du polynôme de degré quatre

$$\psi(\xi) = \delta + \beta\xi + \gamma\xi^2 + \kappa\xi^3 + \iota\xi^4. \tag{B.1}$$

Avant de résoudre ce problème de façon générale, envisageons le cas $\iota = 0$ et $\kappa \neq 0$. Dans ce cas, nous obtenons une équation cubique que nous pouvons résoudre grâce à la méthode de Tartaglia-Cardan (voir par exemple [1]).

Il faut poser

$$q = \frac{\beta}{3\kappa} - \frac{\gamma^2}{9\kappa^2}, \tag{B.2}$$

$$r = \frac{\gamma\beta}{6\kappa^2} - \frac{\delta}{2\kappa} - \frac{\gamma^3}{27\kappa^3}. \tag{B.3}$$

Si $q^3 + r^2 > 0$, il n'y a qu'une racine réelle

$$\xi_3 = \left(r + \sqrt{q^3 + r^2}\right)^{1/3} + \left(r - \sqrt{q^3 + r^2}\right)^{1/3} - \frac{\gamma}{3\kappa}, \tag{B.4}$$

sinon, il y a trois racines réelles

$$\xi_4 = 2\sqrt{-q}\cos\theta - \frac{\gamma}{3\kappa}, \tag{B.5}$$

$$\xi_5 = -\sqrt{-q}\cos\theta - \frac{\gamma}{3\kappa} - \sqrt{-3q}\sin\theta, \tag{B.6}$$

$$\xi_6 = -\sqrt{-q}\cos\theta - \frac{\gamma}{3\kappa} + \sqrt{-3q}\sin\theta \tag{B.7}$$

où

$$\theta = \begin{cases} \frac{1}{3}\arctan\frac{\sqrt{-q^3-r^2}}{r} & \text{si } r \geq 0, \\ \frac{1}{3}\arctan\frac{\sqrt{-q^3-r^2}}{r} + \frac{\pi}{3} & \text{si } r < 0. \end{cases} \tag{B.8}$$

Si $\iota \neq 0$, nous pouvons utiliser la méthode de Ferrari (voir par exemple [1]). Il faut d'abord résoudre le problème cubique en u associé

$$u^3 - \frac{\gamma}{\iota}u^2 + \left(\frac{\beta\kappa}{\iota^2} - 4\frac{\delta}{\iota}\right)u - \left(\frac{\beta^2}{\iota^2} + \frac{\delta\kappa^2}{\iota^3} - 4\frac{\delta\gamma}{\iota^2}\right) = 0 \tag{B.9}$$

par la méthode de Tartaglia-Cardan. S'il n'y a qu'une seule racine u_1, les quatre racines du polynôme (B.1) sont les solutions des deux équations du second degré

$$\begin{aligned} \xi^2 + p_1\xi + q_1 &= 0, \\ \xi^2 + p_2\xi + q_2 &= 0 \end{aligned}$$

où

$$p_1 = \frac{\kappa}{2\iota} - \sqrt{\frac{\kappa^2}{4\iota^2} + u_1 - \frac{\gamma}{\iota}}, \tag{B.10}$$

$$p_2 = \frac{\kappa}{2\iota} + \sqrt{\frac{\kappa^2}{4\iota^2} + u_1 - \frac{\gamma}{\iota}}, \tag{B.11}$$

$$q_1 = \frac{u_1}{2} - \operatorname{sign}\left(\frac{\kappa u_1}{\iota} - 2\frac{\beta}{\iota}\right)\sqrt{\frac{u_1^2}{4} - \frac{\delta}{\iota}}, \tag{B.12}$$

$$q_2 = \frac{u_1}{2} + \operatorname{sign}\left(\frac{\kappa u_1}{\iota} - 2\frac{\beta}{\iota}\right)\sqrt{\frac{u_1^2}{4} - \frac{\delta}{\iota}}. \tag{B.13}$$

S'il y a plus d'une racine réelle au problème (B.9), il convient d'utiliser la valeur de u_1 qui produit des coefficients réels aux deux équations du second degré, i.e. u_1 tel que

$$\frac{\kappa^2}{4\iota^2} + u_1 - \frac{\gamma}{\iota} \geq 0, \tag{B.14}$$

$$\frac{u_1^2}{4} - \frac{\delta}{\iota} \geq 0. \tag{B.15}$$

Bibliographie

[1] Abramowitz, M. Elementary analytical methods. Dans M. Abramowitz et A. Stegun, rédacteurs, *Handbook of mathematical functions*, chapitre 3, pages 10–66. Dover Publications, Inc., New York, seventh Dover édition, 1964.

[2] Alexandre, P. *Algorithmes à métrique variable pour la recherche de zéros d'opérateurs maximaux monotones*. Thèse de doctorat en Sciences, Université de Liège, 1995.

[3] Bischof, C., Carle, A., Khademi, P., et Mauer, A. The ADIFOR 2.0 system for the automatic differentiation of fortran 77 programs. Rapport technique CRPC-TR94491, CRPC, Rice University, 1994.

[4] Boland, N. L. A dual-active-set algorithm for positive semi-definite quadratic programming. *Mathematical Programming*, 78 :1–27, 1997.

[5] Bongartz, I., Conn, A. R., Gould, N., et Toint, P. L. Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21 :123–160, 1995.

[6] Bonnans, J. F., Gilbert, J. C., Lemaréchal, C., et Sagastizabal, C. A family of variable metric proximal methods. Rapport de recherche 1851, INRIA, 1993.

[7] Brohé, M. *Méthodes de type proximal pour une somme d'opérateurs maximaux monotones*. Thèse de doctorat en Sciences, Université de Liège, 2000.

[8] Broyden, C. G. The convergence of a class of double rank minimization algorithms : Parts I and II. *Journal of the Institute of Mathematics and Its Applications*, 6, 1970.

[9] Bruyneel, M. *Schémas d'approximation pour la conception optimale de structures en matériaux composites*. Thèse de doctorat en Sciences Appliquées, Université de Liège, 2002.

[10] Bruyneel, M., Vermaut, O., et Fleury, C. Reliable approximation schemes for composite structures optimization. Dans R. Van Keer, B. Verhegghe,

M. Hogge, et E. Noldus, rédacteurs, *International Conference on Advanced Computational Methods in Engineering ACOMEN98*, pages 705–712. Shaker Publishing B.V., Ghent, Belgium, 1998.

[11] Bunch, J. R. et Kaufman, L. A Computational Method for the Indefinite Quadratic Programming Problem. *Linear Algebra and Its Applications*, 34 :341–370, 1980.

[12] Byrd, R. H., Khalfan, H. F., et Schnabel, R. B. Analysis of a symmetric rank-one trust region method. *SIAM Journal on Optimization*, 6(4) :1025–1039, 1996.

[13] Cartis, C., Gould, N., et Toint, P. L. Adaptive cubic overestimation methods for unconstrained optimization. Part I : motivation, convergence and numerical results. *Mathematical Programming A*, (to appear).

[14] Cartis, C., Gould, N., et Toint, P. L. Adaptive cubic overestimation methods for unconstrained optimization. Part II : worst-case function-evaluation complexity. *Mathematical Programming A*, (to appear).

[15] Charles, J. F., Habraken, A. M., et Lecomte, J. Modelling of elasto-visco-plastic behaviour of steels at high temperatures. Dans J. Huétink et F. P. T. Baaijens, rédacteurs, *NUMIFORM 98, Simulation of Materials Processing : Theory, Methods and Applications*, pages 277–282. A.A.Balkema, Enschede, The Netherlands, 22–25 June 1998.

[16] Chen, G. et Teboulle, M. Convergence analysis of proximal-like minimization algorithm usint Bregman functions. *SIAM Journal on Optimization*, 3(3) :538–543, 1993.

[17] Cohen, G. Auxiliary problem principle and decomposition of optimization problems. *Journal of Optimization Theory Applications*, 32(3) :277–305, 1980.

[18] Cohen, G. Auxiliary problem principle extended to variational inequalities. *Journal of Optimization Theory Applications*, 59(2) :325–334, 1988.

[19] Conn, A. R., Gould, N. I. M., et Toint, P. L. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50(2) :177–195, 1991.

[20] Conn, A. R., Gould, N. I. M., et Toint, P. L. *Trust-Region Methods*. MPS/SIAM Series on Optimization. SIAM, Philadelphia, USA, 2000.

[21] Davidon, W. C. Variable metric method for minimization. U.S. Atomic Energy Commission Research and Developement Report ANL-5990, Argonne National Laboratories, 1959.

[22] Davidon, W. C. Conic approximations and collinear scalings for optimizers. *SIAM Journal on Numerical Analysis*, 17(2) :268–281, 1980.

[23] Dennis, J. E. et Mei, H. H. W. Two new unconstrained optimization algorithms which use function and gradient values. *Journal of Optimization Theory and Applications*, 28(4) :453–482, 1979.

[24] Dolan, E. D. et Moré, J. J. Benchmarking optimization software with performance profiles. *Mathematical Programming, Series A*, 91(2) :201–213, 2002.

[25] Duysinx, P., Zhang, W. H., Fleury, C., Nguyen, V. H., et Haubruge, S. A new separable approximation scheme for topological problems and optimization problems characterized by a large number of design variables. Dans N. Olhoff et G. I. N. Rozvany, rédacteurs, *First World Congress of Structural and Multidisciplinary Optimization*, pages 1–8. ISSMO, Goslar, Germany, 1995.

[26] Eckstein, J. Nonlinear proximal point algorithm using bregman functions. *Mathematics of operations research*, 18(1) :202–226, 1993.

[27] Faure, C. An automatic differentiation platform : Odyssée. *Future Generation Computer Systems*, 21(8) :1391–1400, 2005.

[28] Fletcher, R. A new approach to variable metric algorithms. *Computer Journal*, 13 :317–322, 1970.

[29] Fletcher, R. A General Quadratic Programming Algorithm. *Journal of the Institute of Mathematics and Its Applications*, 7(1) :76–91, 1971.

[30] Fletcher, R. *Practical methods of optimization*, tome 1 : Unconstrained optimization. John Wiley & Sons, 1980.

[31] Fletcher, R. *Practical methods of optimization*. John Wiley & Sons, second édition, 1987.

[32] Fletcher, R., Gould, N. I. M., Leyffer, S., Toint, P. L., et Wächter, A. Global convergence of a trust-region sqp-filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3) :635–659, 2002.

[33] Fletcher, R. et Leyffer, S. Nonlinear programming without a penalty function. *Mathematical Programming, Series A*, 91(2) :239–269, 2002.

[34] Fletcher, R. et Powell, M. J. D. A rapidly convergent descent method for minimization. *Computer Journal*, 6 :163–168, 1963.

[35] Fleury, C. Efficient approximation concepts using second order information. *International Journal for Numerical Methods in Engineering*, 28(9) :2041–2058, 1989.

[36] Fleury, C. et Braibant, V. Structural optimisation : a new dual method using mixed variables. *International Journal for Numerical Methods in Engineering*, 23 :409–428, 1986.

[37] Fukushima, M., Luo, Z.-Q., et Tseng, P. A sequential quadratically constrained quadratic programming method for differentiable convex minimization. *SIAM Journal on Optimization*, 13(4) :1098–1119, 2003.

[38] Giering, R. et Kaminski, T. Applying TAF to generate efficient derivative code of fortran 77-95 programs. *PAMM*, 2(1) :54–57, 2003.

[39] Gilbert, J. C. et Lemaréchal, C. Some numerical experiments with variable-storage quasi-newton algorithms. *Mathematical Programming*, 45 :407–435, 1989.

[40] Gilbert, J. C. et Lemaréchal, C. *The modules M1QN3 and N1QN3, Version 2.0c*. INRIA, B.P. 105, 78153 Le Chesnay Cedex, France, June 1995.

[41] Gill, P. E., Murray, W., et Saunders, M. A. SNOPT : an SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1) :99–131, 2005.

[42] Gill, P. H. et Murray, W. Numerically stable methods for quadratic programming. *Mathematical Programming*, 14 :349–372, 1978.

[43] Goldfarb, D. A family of variable metric methods derived by variational means. *Mathematics of Computation*, 24 :23–26, 1970.

[44] Gould, N., Lucidi, S., Roma, M., et Toint, P. L. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization*, 9(2) :504–525, 1999.

[45] Gould, N. I. M., Orban, D., Sartenaer, A., et Toint, P. L. Sensitivity of trust-region algorithms to their parameters. Rapport technique 04/07, Optimization Online Digest, August 2004.

[46] Gould, N. I. M., Orban, D., Sartenaer, A., et Toint, P. L. Sensitivity of trust-region algorithms to their parameters. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 3(3) :227–241, 2005.

[47] Gould, N. I. M., Orban, D., et Toint, P. L. CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, revisited. *Transactions of the American Mathematical Society on Mathematical Software*, 29(4) :373–394, 2003.

[48] Griewank, A., Juedes, D., et Utke, J. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Transactions on Mathematical Software*, 22(2) :131–167, 1996.

[49] Hascoët, L. et Pascual, V. TAPENADE 2.1 user's guide. Rapport technique 0300, INRIA, Sophia, Antipolis, 2004.

[50] Hei, L. A self-adaptive trust region algorithm. *Journal of Computational Mathematics*, 21(2) :229–236, 2003.

[51] Jeunechamps, P.-P., Duysinx, P., Walmag, J. M. B., Mathonet, V., Delhez, É. J. M., Tossings, P., Habraken, A. M., et Ponthot, J.-P. A trust-region algorithm for automatic identification of elasto-viscoplastic parameters in metal forming problems. Dans J. Kusiak et M. Pietrzyk, rédacteurs, *Proceedings of the 10th International Conference on Metal Forming 2004, Steel Grips — Journal of steel and related materials*, tome 2, pages 527–534. Akademia Górniczo-Hutnicza, Kraków, Poland, 19–22 September 2004.

[52] Jeunechamps, P.-P., Walmag, J. M. B., Mathonet, V., Delhez, É. J. M., Habraken, A. M., Ponthot, J.-P., Tossings, P., et Duysinx, P. Identification of elastoplastic model parameters in large deformation problems. Dans *The 5th World Congress of Structural and Multidisciplinary Optimization*. ISSMO, Italian Polytechnic Press, Lido di Jesolo, Italy, 2005.

[53] Jian, J. B. A quadratically approximate framework for constrained optimization, global and local convergence. *Acta Mathematica Sinica, English Series*, 24(5) :771—-788, 2008.

[54] Jian, J. B., Hu, Q. J., Tang, C. M., et Zheng, H. Y. A sequential quadratically constrained quadratic programming method of feasible directions. *Applied Mathematics and Optimization*, 56(3) :343–363, 2007.

[55] Kabbadj, S. *Méthodes proximales entropiques*. Thèse de doctorat, Université de Montpellier II, 1994.

[56] Keller, E. L. The general quadratic optimization problem. *Mathematical Programming*, 5 :311–317, 1973.

[57] Kleinermann, J.-P. *Identification paramétrique et optimisation de mise à forme par problèmes inverses*. Thèse de doctorat en Sciences Appliquées, Université de Liège, 2000.

[58] Kruk, S. et Wolkowicz, H. SQ^2P, sequential quadratic constrained quadratic programming. Dans Y. Yuan, rédacteur, *Advances in Nonlinear Programming*, pages 177–204. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[59] Kuhn, H. W. et Tucker, A. W. Nonlinear programming. Dans J. Neyman, rédacteur, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, Berkeley, 1951.

[60] Lawson, L. M., Hoffman, E. E., et Spitz, Y. H. Time series sampling and data assimilation in a simple marine ecosystem model. *Deap-Sea Research II*, 43(2–3) :625–651, 1996.

[61] Lawson, L. M., Spitz, Y. H., Hofmann, E. E., et Long, R. B. A data assimilation technique applied to a predator-prey model. *Bulletin of Mathematical Biology*, 57 :593–617, 1995.

[62] Lehoucq, R. B. The computation of elementary unitary matrices. *ACM Transactions on Mathematical Software*, 22(4) :393–400, 1996.

[63] Lemaire, B. Coupling optimization methods and variational convergence. Dans K. H. Hoffmann, J. B. Hiriart-Urruty, C. Lemaréchal, et J. Zowe, rédacteurs, *Trends in Mathematical Optimization*, tome 84 de *International Series of Numerical Mathematics*, pages 163–179. Birkhäuser Verlag, Basel, 1988.

[64] Lemaréchal, C. et Mifflin, R. Global and superlinear convergence of an algorithm for one-dimensional minimization of convex functions. *Mathematical Programming*, 24 :241–256, 1982.

[65] Lemaréchal, C., Oustry, F., et Sagastizábal, C. The $\mathcal{U}$-Lagrangian of a convex function. *Transactions of the American Mathematical Society*, 352 :711–729, 1999.

[66] Leredde, Y., Devenon, J.-L., et Dekeyser, I. Peut-on optimiser les constantes d'un modèle de turbulence marine par assimilation d'observations ? *Comptes Rendus de l'Académie des Sciences — Series IIA — Earth and Planetary Sciences*, 331(6) :405–412, 2000.

[67] Litt, F.-X. Analyse numérique II. Université de Liège, Notes de cours, 1999.

[68] Litt, F.-X. Introduction à la théorie de l'optimisation. Université de Liège, Notes de cours, 2001.

[69] Maratos, N. *Exact penalty function algorithm for finite dimensional and control optimization problems*. Thèse de doctorat, University of London, 1978.

[70] Martinet, B. *Algorithmes pour la résolution de problèmes d'optimisation et de minimax*. Thèse d'État, Université de Grenoble, 1972.

[71] Matear, R. J. Parameter optimization and analysis of ecosystem models using simulated annealing : A case study at station P. *Journal of Marine Research*, 53 :571–607, 1995.

[72] Michaleris, P., Tortorelli, D. A., et Vidal, C. A. Tangent operators and design sensitivities for transient non-linear coupled problems with application to elastoplasticity. *International Journal for Numerical Methods in Engineering*, 37(14) :2471–2499, 1994.

[73] Mifflin, R. et Sagastizábal, C. $\mathcal{V}\mathcal{U}$-decomposition derivatives for convex max-functions. Dans R. Tichatschke et M. Théra, rédacteurs, *Ill-posed Variational Problems and Regularization Techniques*, numéro 477 dans Lecture Notes in Economics and Mathematical Systems, pages 167–186. Springer-Verlag, Berlin Heidelberg, 1999.

[74] Moré, J. M. et Sorensen, D. C. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3) :553–572, 1983.

[75] Murray, J. D. *Mathematical Biology. I : An Introduction*, tome 17 de *Interdisciplinary Applied Mathematics*. Springer, third édition, 2002.

[76] Murray, W. et Overton, M. L. Steplength algorithms for minimizing a class of nondifferentiable functions. *Computing*, 23 :309–331, 1979.

[77] Navon, I. M. Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography. *Dynamics of Atmospheres and Oceans*, 27 :55–79, 1997.

[78] Nihoul, J. C. J. introduction à l'étude de la turbulence et à la modélisation des fluides géophysiques. A Modelenvironment-Interscience Publication, 1997.

[79] Nihoul, J. C. J. Systèmes non-linéaires. Université de Liège, Notes de cours, 1999.

[80] Nocedal, J. et Wright, S. J. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.

[81] Panier, E. R. An active set method for solving linearly constrained nonsmooth optimization problems. *Mathematical Programming*, 37 :269–292, 1987.

[82] Papalambros, P. Y. et Wilde, D. J. *Principles of optimal design*. Cambridge University Press, second édition, 2000.

[83] Pirlot, M. General local search heuristics in combinatorial optimization : a tutorial. *Belgian Journal of Operations Research, Statistics and Computer Science*, 32(1–2), 1992.

[84] Powell, M. J. D. Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, 28 :487–500, 1986.

[85] Press, W. H., Teukolsky, S. A., Vetterling, W. T., et Flannery, B. P. *Numerical Recipes in Fortran 77 : The Art of Scientific Computing*, tome 1 de *Fortran Numerical Recipes*, chapitre 10, pages 387–448. Press Syndicate of the University of Cambridge, second édition, 1986.

[86] Pryce, J. D. et Reid, J. K. AD01, a fortran 90 code for automatic differentiation. Rapport technique RAL-TR-1998-057, Rutherford Appleton Laboratory, Chilton, Didcot, Oxfordshire OX11 0QX, England, 1998.

[87] Remouchamps, A. et Radovcic, Y. Boss/Quattro : Theoritical aspects about optimisation methods and algorithms. Rapport technique, Samtech s.a., 2001.

[88] Renaud, A. *Algorithmes de régularisation et décomposition pour les problèmes variationnels monotones*. Thèse de doctorat, E.N.S. des Mines de Paris, 1993.

[89] Rockafellar, R. T. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5) :877–898, 1976.

[90] Schittkowski, K. *Numerical data fitting in dynamical systems*, tome 77 de *Applied Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[91] Serres, M. et Farouki, N., rédacteurs. *Le Trésor, Dictionnaire des Sciences*. Flammarion, Paris, 1997.

[92] Shanno, D. F. Conditionning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24 :647–656, 1970.

[93] Spitz, Y. H. *A feasibility study of dynamical assimilation of tide gauge data in the Chesapeake Bay*. Thèse de doctorat, Old Dominion University, 1995.

[94] Spitz, Y. H., Moisan, J. R., Abott, M. R., et Richman, J. G. Data assimilation and a pelagic ecosystem model : parametrization using time series observations. *Journal of Marine Systems*, 16 :51–68, 1998.

[95] Stoer, J. et Bulirsch, R. *Introduction to numerical analysis*. Numéro 12 dans Texts in applied mathematics. Springer-Verlag, New York, third édition, 2002.

[96] Svanberg, K. The method of moving asymptotes — a new method for structural optimization. *International journal for numerical methods in engineering*, 24 :359–373, 1987.

[97] Svanberg, K. A globally convergent version of mma without linesearch. Dans N. Olhoff et G. I. N. Rozvany, rédacteurs, *First World Congress of Structural and Multidisciplinary Optimization*, pages 9–16. ISSMO, Goslar, Germany, 1995.

[98] Tossings, P. *Sur les zéros des opérateurs maximaux monotones et applications*. Thèse de doctorat en Sciences, Université de Liège, 1990.

[99] Walmag, J. M. B. *Optimisation des paramètres d'un modèle dynamique d'écosystème par assimilation de données*. Travail de fin d'études, Université de Liège, 2002.

[100] Walmag, J. M. B. et Delhez, É. J. M. A note on trust-region radius update. *SIAM Journal on Optimization*, 16(2) :548–562, 2005.

[101] Walmag, J. M. B. et Delhez, É. J. M. A trust-region method applied to parameter identification of a simple prey-predator model. *Applied Mathematical Modelling*, 29(3) :289–307, 2005.

[102] Wächter, A. et Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1) :25–57, 2006.

[103] Zhang, W. H. et Fleury, C. A modification of convex approximation methods for structural optimization. *Computers & Structures*, 64(1) :89–95, 1997.

[104] Zheng, C. et Wang, P. Parameter structure identification using tabu search and simulated annealing. *Advances in Water Resources*, 19(4) :215–224, 1996.